明 細 書

データ処理装置

技術分野

[0001] 本発明は、再構成可能な論理回路領域を有するデータ処理装置に関するものである。

背景技術

[0002] 回路を再構成できるプログラマブルデバイスとして、FPGA(Field Programmable Gate Array)、PLD(Programmable Logic Device)、PLA(Programmable Logic Array)と称されるデバイスが知られている。これらのプログラマブルデバイスの基本的な構成は、論理セルあるいは論理ユニットと称されるユニットが格子状に配置され、それを取り巻くように配線群が配置されたものであり、コンテキスト情報あるいはコンフィグレーション情報と呼ばれる情報によって、論理セルの機能や配線の接続を変更できるようになっている。

[0003] 特開2000−40745号公報には、FPGAに論理回路の異なる部分を実装する技術の1つとして、論理回路を特徴付ける初期ネットリストを多くのページへと区分し、FPGAにこれらのページの1つの回路を実装することが記載されている。これにより、FPGAの物理的容量よりもはるかに大きな回路の実装を可能にしようとしている。

[0004] しかしながら、現在、マルチメディアデバイス、モバイルデバイス、デジタルデバイスなどに搭載され、それらのデバイスのデータ処理の多くを行っているシステムLSIは、1つのチップの上に、特定の機能を実現するための回路単位(多くのケースでは、ハードウェアモジュールあるいはIP(Intellectual Property)、ライブラリと称される)が複数搭載され、それらのハードウェアモジュールが並列して処理を行っている。したがって、FPGAに、単に1つの回路を分解して実装したとしても、回路を再構成可能なデバイスの有効性が大きく広がることにはならない。

発明の開示

[0005] これに対し、本発明においては、アプリケーションを実行するために、または、アプリケーションを実行する際に、ハードウェア空間を動的に最適化する技術を提供する。

そして、コンパイラ翻訳による命令セットの集合であるプログラムのような、従来のソフトウェア情報だけでは無く、ハードウェア情報であるアプリケーションを実行する回路そのものの情報を得て、それを直接実行可能とするアーキテクチャを備えたデータ処理装置、例えば、LSIを本発明においては提供する。

[0006] 従来のシステムLSIに代表される回路デバイスの技術においては、ハードウェアで実現する回路は、特定のハードウェアモジュール、IP、ライブラリと呼ばれる単位で、固定されたハードウェアとして搭載され、それらの回路により専用化されたLSIによりデータが処理される。これに対し、汎用的な回路あるいはアーキテクチャでハードウェアの回路を実行する技術がある。例えば、シミュレータであり、プログラムの命令を1つ1つ実行して、その回路自体を実行しているかのように、汎用的な装置であるCPUに役割を割り当てる手法である。これは、本来、ハードウェアで実現する回路の持つ並列性を命令単位の実行に変えてCPUで処理を行うため、複数のCPUを使用したとしても、実際のハードウェアの回路と比較して、実行する回路規模にもよるが、普通3桁ー5桁以上の実行時間が必要とされる。また、リアルタイム性に決定的に欠けるので、実際に複雑な回路をシミュレーションしようとすると膨大な検証時間を必要とする。このため、膨大なゲート数を有する近年の専用LSIに代わり処理を実行することが不可能なだけではなく、複雑な専用LSIの機能を検証するにも不向きな状況になりつつある。

[0007] リアルタイム性の問題に対して、ハードウェア・アクセラレーションと呼ばれる手法がある。これは、最初の段階では、CPUやDSPを複数並べて並列実行させ、その一つ一つに小規模な回路を割り当てて、全体としてはシミュレーション時間を短時間で行うというアイデアである。FPGAやPLDが実用化されたことにより、シミュレーション対象の回路を直接これに割り当てるやり方が主流となり、大規模な集積回路やハードウェアのエミュレーションが、リアルタイムに極めて近い時間で実行できるようになりつつある。

[0008] しかしながら、FPGAの内部セル構造は、ハードウェア回路の実現をある一定の時間を掛けて変更するようなアーキテクチャ構造となっており、CPUやDSP等のデータパス系を有するハードウェアの実装にはあまり向いていない。実際に、実装しようとす

ると、処理性能(動作周波数)・ゲート効率・消費電力の何れでも、専用に設計されたLSIと競争できない。さらに、アプリケーションを実行するための回路にもよるが、FPGAまたはPLDの場合、実装対象となる回路の規模がFPGAあるいはPLDの集積度より大きいと、原則として実装は不可能である。アプリケーションを実行するための回路を分割して実装する事も考えられるが、その場合、チップが複数になりピン数の制限から、更に性能・コスト・消費電力とも不利になる。

[0009] さらに、1つのFPGAに分割した回路を実装する場合、ピン数の制限や、分割した回路同士の境界情報の伝播など、回路を分割する際に発生する問題が多数ある。それらを、FPGAを用いたハードウェアの設計の段階ですべて解決しようとすると、専用LSIを設計および開発する従来の技術に対し、FPGAを用いたメリットは失われてしまう。さらに、一般的に、FPGAやPLDは、目標のハードウェア回路の数倍ー数十倍ものハードウェアが必要となり、チップ・コスト、目標性能、および消費電力の3つの点でも、専用LSIには及ばない。

[0010] 一方、専用LSIにも多くの問題がある。従来の専用LSIの場合、LSIの設計段階で正確な性能目標や機能仕様が無いと設計が収束しない。例えば、アプリケーションの実行状況によっては、機能と性能の動的トレード・オフが可能なことが多いが、設計段階で必要とされる性能を実現可能なだけのハードウェア領域や性能を保証出来る動作周波数を確定させる必要がある。つまり、機能や性能の動的トレード・オフがアプリケーション上可能な場合でも、ピーク性能要求や単体レベルの機能毎の性能保証を行った上で、LSI設計を行う必要がある。従って、機能と性能についての要求が決定的であり、高性能および多機能を狙うシステムLSIは、コスト的には最悪ケースの積み上げとなり、製造コスト、チップ面積、消費電力などが悪化する一方になる。

[0011] さらに具体的には、自動制御装置、例えば、ロボットのようなアプリケーションの場合、視覚情報や聴覚情報を処理しているときは、他の機能(歩行機能・言語処理・嗅覚処理等)を大幅に弱めて良い場合が多い。しかしながら、従来のシステムLSIでは、すべての機能を実現するすべての回路を同じようにシステムLSIに実装しており、単にその処理結果を使用しないか、あるいは、スタンバイ状態で処理能力を低下させているに過ぎない。

[0012] これに対し、回路構成を動的に再構成できるのであれば、その再構成可能な論理回路により構成されるハードウェア空間を動的に最適化することにより、使用しない、あるいはスタンバイ状態になる機能へのハードウェア資源の割り当てを大幅に絞って、本来集中すべき視覚情報処理や聴覚情報処理にハードウェア資源を集中的に割り当てることができる。すなわち、本発明によれば、従来の専用LSIのように、ハードウェア回路(ターゲット回路)をすべて実装する必要がないので、少ないハードウェア資源で最大の実行効率を得ることができる。

[0013] 本発明における動的最適化技術は、論理回路により構成される実ハードウェア空間のアレンジを動的に最適化することを言い、実ハードウェア空間をその都度刷新するだけではなく、実ハードウェア空間の部分的なアレンジを動的に最適化することも含む概念である。したがって、現在使用していない機能へのハードウェア資源の割り当てを無くすだけではなく、ハードウェア資源の割り当てを絞り、スタンバイ中の機能のリアルタイム応答性を犠牲にすることなく、使用中の機能に対するハードウェア資源の割り当てを増加させることができるものである。

[0014] また、本発明における動的最適化技術は、使用中であっても緊急性を要する機能に対してはハードウェア資源の割り当てを増加し、緊急性を要しない機能に対してはハードウェア資源の割り当てを減らしたり、ハードウェア資源の割り当てを一時的に無くすことができるものである。緊急性とは、処理速度、優先順位などを含む概念であり、データ処理装置に対する要求の重要なものの1つである。ハードウェア資源の割り当てを左右する、データ処理装置に対するその他の要求としては、並列処理するジョブの増減、割り込みの有無など様々なものが考えられる。本発明における動的最適化技術は、これらのデータ処理装置に対する要求に応じてハードウェア資源の割り当て、すなわち、実ハードウェア空間の構成を動的に最適化する。

[0015] 実ハードウェア空間を動的に最適化する1つの方法は、データ処理装置が遭遇するすべての場面を想定し、それに対して最適な実ハードウェア空間のアレンジを予め決定し、コンテキスト情報(あるいはコンフィグレーション情報)として用意し、その都度、ロードする方法である。この方法は、実ハードウェア空間で生ずるタイミング収束などの問題を予め解決できるので、データ処理装置の性能を確保する点では望ましい

であろう。しかしながら、すべてのシナリオにおいて遭遇する場面を想定することは不可能であり、限られた場面を対象としてある程度最適化し、その他の場面では中庸な性能が得られるような汎用的な解を得ようとすれば、実ハードウェア空間を動的に最適化する効果は薄れてしまう。

[0016] 他の方法の1つは、ある機能を実装するための単位として設計されるハードウェアモジュール(IPまたはライブラリ)を、論理ゲートとそれらの接続状態を示しただけのネットリストの状態で用意し、その機能が必要となったときに、そのネットリストの一部あるいは全体を実ハードウェア空間の空いた空間にあわせて動的に配置および配線し、マッピングする方法である。この方法は、実ハードウェア空間の瞬間的な状況に合わせて回路をフレキシブルに、動的に配置できるので、ハードウェア空間を最も動的に最適化できる方法であると考えられる。しかしながら、ネットリストに基づく配置および配線する処理は、静的なLSIの設計および開発段階でも膨大な時間を要する処理であり、その処理をマッピングする瞬間毎に繰り返す必要がある。瞬間的な実ハードウェア空間の実情とその他の要素を加味し、瞬間的に必要な回路のネットリストに基づいて配置・配線の諸問題を解決してマッピングすることは実際には不可能である。ほとんどクロック単位あるいはサイクル単位で高速にタイミング収束を含めた問題を解決できるハードウェアが提供できたとしても、そのようなハードウェア資源を別途用意することは経済的でもないし、そのようなハードウェアの登場を待つのでは、ハードウェア空間の動的最適化の実現が難しくなるだけである。

[0017] ネットリストの状態から、そのネットリストで実装しようとしている回路の一部の適当な範囲を単位とし、それぞれの回路単位内の配置・配線を解決し、それらの回路単位を実ハードウェア空間の空いた空間に配置して、それらの回路単位を接続する配置・配線問題に縮小することにより、マッピングする瞬間毎の配置・配線問題を解決する時間を短縮できる可能性がある。しかしながら、配置・配線するときの実ハードウェア空間の状況は刻々と変動する。したがって、状況に応じて常に瞬間瞬間で配置・配線問題を動的に解決することは容易ではないであろうし、可能であったとしてもハードウェア資源と電力をそのために常に費やすことには変わりなく、高性能高機能・低チップコスト・低消費電力のデータ処理装置を提供するという課題を解決することができ

ない。

[0018] そこで、本発明においては、アプリケーションを実行するための回路の少なくとも一部であるオブジェクト回路を論理回路領域の一部にマッピングするためのオブジェクト回路情報と、オブジェクト回路に接するインタフェース回路を論理回路領域にマッピングするためのインタフェース回路情報と、インタフェース回路において実現する境界条件とを備えたアーキテクチャコードを提供する。オブジェクト回路の最も適当な例は、アプリケーションを実行するためのある機能を実装するための単位として設計されるハードウェアモジュールを複数に分割した分割回路である。ハードウェアモジュールを実装するために要するハードウェア資源が少ない場合は、分割せずにオブジェクト回路化し、それに対応するインタフェース回路情報を生成して論理回路領域に実装することも可能である。

[0019] 回路を動的に再構成可能な論理回路領域を有するデータ処理装置における、本発明の制御方法は、アーキテクチャコードを取得する工程と、アーキテクチャコードのオブジェクト回路情報およびインタフェース回路情報により、論理回路領域にオブジェクト回路と、そのオブジェクト回路に接するインタフェース回路とをマッピングする工程と、アーキテクチャコードの境界条件に基づきインタフェース回路を制御する動作工程とを有する。

[0020] また、本発明のデータ処理装置は、回路を動的に再構成可能な論理回路領域と、アーキテクチャコードを取得するロードユニットと、アーキテクチャコードのオブジェクト回路情報およびインタフェース回路情報により、論理回路領域にオブジェクト回路と、そのオブジェクト回路に接するインタフェース回路とをマッピングするマッピングユニットと、アーキテクチャコードの境界条件にしたがってインタフェース回路を制御する動作制御ユニットとを有する。ロードユニットは、アーキテクチャコードをフェッチする場合はフェッチユニットであり、コンピュータなどを用いた通信ネットワークやメモリからダウンロードする場合はダウンロードユニットになる。ロードした回路情報によりハードウェアを再構成するマッピングを含めてロードと呼ばれることもあるが、本明細書においては、アーキテクチャコードを取得するまでをロードするステップと呼ぶことにする。ロードユニットにおいてコードを取得するプロセスには、フェッチ、ダウンロード、ゲット、

リードなど様々な命令を割り当てることができ、コミュニケーションシステムによりアーキテクチャコードをロードすることも可能である。

[0021] これらのロードユニット、マッピングユニットおよび動作制御ユニットも、ハードウェアモジュールの1つとして捉えることが可能であり、分割してオブジェクト回路化することにより、論理回路領域に分割回路として実装することが可能である。したがって、アプリケーションの実行状況により、ロードユニット、マッピングユニットおよび動作制御ユニットの機能の一部を実現すれば良いような条件においては、これらの機能に割かれるハードウェア資源を解放して、他のハードウェアモジュールの実現のためにハードウェア資源を割当てて並列処理能力を向上したり、処理速度を改善したりすることができる。

[0022] 本発明においては、ハードウェアモジュールを含むアプリケーションを実行するための回路のオリジナルのネットリストから、ある適当な範囲を分割し、分割されたユニット内で配置・配線問題が解決され、論理回路領域の一部にマッピング可能なオブジェクト回路情報を生成する。次に、オブジェクト回路情報により構成されるオブジェクト回路の、オリジナルのネットリスト上の境界を形成する情報からインタフェース回路情報を生成する。さらに、オリジナルのネットリストをオブジェクト回路の集合に変換して、それらのオブジェクト回路の間の配置・配線問題を解決し、インタフェース回路における境界条件を生成する。したがって、オブジェクト回路の間の配置・配線問題は、インタフェース回路における境界条件として、アーキテクチャコードの生成段階で解決される。

[0023] このため、実ハードウェア空間である、回路を動的に再構成可能な論理回路領域の空いた空間に、オブジェクト回路を配置するときには、ロードユニットにより、適切なアーキテクチャコードを取得し、マッピングユニットによりオブジェクト回路をマッピングすると共に、その周囲にインタフェース回路をマッピングし、実行制御ユニットにより、インタフェース回路を境界条件に基づいて制御することにより、実ハードウェア空間にオブジェクト回路を動的に配置し、オブジェクト回路を実行することができる。したがって、実ハードウェア空間の瞬間的な状態により、所望の、あるいは適切なオブジェクト回路とインタフェース回路を論理回路領域にアレンジするだけで、オブジェクト回路

を実行することができる。そして、瞬間瞬間でオブジェクト回路の間の配置および配線問題を解決しなければならないという問題の発生を未然に防止できる。

[0024] 本発明においては、実ハードウェア空間である再構成可能な論理回路領域にマッピングされた1つまたは複数のオブジェクト回路は、その状態で実行される。オブジェクト回路の境界はインタフェース回路を介して、仮想的には、そのオブジェクト回路が属するハードウェアモジュールを構成する多数のオブジェクト回路に接続されている。このため、オブジェクト回路の境界は多数のオブジェクト回路が接続された仮想ハードウェア空間の状態で制御される。したがって、マッピングする際は、オブジェクト回路とインタフェース回路とを、実ハードウェア空間である論理回路領域の利用可能ないずれの領域にもマッピングすることができる。

[0025] さらに、隣り合うオブジェクト回路との境界におけるインタフェース回路情報および境界条件が一致する、または対応している場合は、マッピングしようとしているオブジェクト回路と隣り合うオブジェクト回路とは仮想ハードウェア空間において隣接しているオブジェクト回路であることを意味する。このため、インタフェース回路を経ずに隣り合うオブジェクト回路同士が直に接続されるようにオブジェクト回路をマッピングすることができる。すなわち、実ハードウェア空間にマッピングする実際のオブジェクト回路の集合の大きさを、実ハードウェア空間の状態に合わせて自由に変えることができる。複数のオブジェクト回路を実ハードウェア空間に分散してマッピングすることも、集中してマッピングすることも可能であり、実ハードウェア空間を極めてフレキシブルに使用することができる。

[0026] 論理回路領域に現在および／または過去に、時間的および／または空間的に分散配置されたオブジェクト回路同士の接続も容易である。時間的および／または空間的に分かれてマッピングされた他方のオブジェクト回路のインタフェース回路の状態を、動作制御ユニットにより、境界条件に基づき、実行する一方のオブジェクト回路のインタフェース回路の制御に反映することにより、実ハードウェア空間では時間的あるいは空間的に分かれたオブジェクト回路同士を、仮想ハードウェア空間では無理なく接続することができる。このため、論理回路領域に現在および／または過去にマッピングされたオブジェクト回路のインタフェース回路の状態を記憶する境界情報メモ

リを設けておくことが望ましい。他方のオブジェクト回路には、時間的および／または空間的に接続されずにマッピングされた自己のオブジェクト回路も含まれる。これにより、あるオブジェクト回路を条件によって実ハードウェア空間では消滅・再生しても、仮想ハードウェア空間で連続した処理が可能である。さらに、同一のオブジェクト回路情報による回路インスタンスを複数構成して並列度を高めたり、信頼度を高めたりすることも可能である。そのような制御はアーキテクチャコードによっても、マッピングユニットと動作制御ユニットの組合せによっても可能である。

[0027] 本発明のアーキテクチャコードは、様々な利用方法がある。実ハードウェア空間にマッピングするアーキテクチャコードをプログラムの命令セットのようにシーケンシャルにトレースできる状態で提供することにより、アーキテクチャコードによりデータ処理装置を制御できる。アーキテクチャコードは、記録媒体に記録して提供することも可能であるし、ネットワークなどの通信手段を介して提供することも可能であり、ハードウェアの構成を遠隔操作により変えることも可能である。

[0028] また、従来のシステムLSIに代わる使用方法としては、ロードユニットにより、データ処理装置に対する要求（緊急性や、新たなジョブの開始あるいは並列処理状態の変化、割り込みの有無など）、マッピングされたオブジェクト回路の実行状況、論理回路領域の利用可能状況を含めた動作環境情報に基づき、複数のアーキテクチャコードを備えたアーキテクチャライブラリから所望のアーキテクチャコードを取得し、アプリケーションの実行状況によりデータ処理装置のハードウェアを動的に最適な構成にすることができる。最適化の指針は、動作環境情報に基づき決めることができ、それにはロードユニット、マッピングユニット、実行制御ユニットの1つまたは複数が寄与する。実ハードウェア空間の空いた空間に新たなオブジェクト回路をマッピングしたり、使用済みのオブジェクト回路を消去したり、緊急性を有するハードウェアモジュールを優先的にマッピングするために、他のハードウェアモジュールの分割回路を一時的に退避したり、他のハードウェアモジュールに割り当てられたハードウェア資源の割合を一時的に縮小したり、実ハードウェア空間の利用方法に制限はない。

[0029] また、ハードウェア回路をアーキテクチャコード化することにより、データ処理装置の利用価値は飛躍的に増大する。限られた利用方法としては、アーキテクチャライブラ

リをデータ処理装置に実装することができる。例えば、このデータ処理装置と、データ処理装置に実装される少なくとも1つのハードウェアモジュールが、それぞれの制御またはデータ処理に用いられる複数の自動制御機構とを有し、論理回路領域に、複数のハードウェアモジュールの分割回路がそれぞれ動的にマッピングされる自動制御装置を提供できる。ロボットの視覚情報や聴覚情報を処理しているときは、それらの制御に必要なハードウェアモジュールの分割回路にハードウェアリソースが主に割り当てられ、歩行機能などの他の自動制御機構の制御に必要なハードウェアモジュールの分割回路は実ハードウェア空間から追い出されるといった制御が可能となる。

[0030] 一方、ネットワークなどのデータ処理装置の外側に対する通信を介してアーキテクチャコードを取得することも可能であり、オブジェクト回路をマッピングできる程度のハードウェア空間があれば、膨大なハードウェア資産を自由に利用することができる。たとえばインターネット上に存在する多種多様なハードウェア資産を手元の携帯端末のLSIにマッピングして利用することが可能となる。ロボットのように、種々の機構を備えた装置と組み合わせることにより、小さなリソースで多種多様な作業を行うことができる。また、アーキテクチャコードは常にアップデートされるので、常にアップデートされた制御回路の下で多種多様な作業を行わせることができる。

[0031] また、本発明のデータ処理装置と、アンテナ、コネクタなどの、外界との通信のために回路では実現できない通信に必要な機構と、ディスプレイ、マイクなどの入出力に特化した機構とを備えた端末により、種々の通信を行う機能から、通信以外の機能、例えば、身体の状態をモニタする機能など、様々なサービスを提供できる。そして、それぞれのサービスを行う回路構成は、常にアップデートされた最新なものを使用できる。

[0032] オブジェクト回路のサイズはフレキシブルであり、マッピング対象の論理回路領域にインタフェース回路も含めてマッピングできるサイズよりも小さければ良い。オブジェクト回路のサイズは小さい方が実ハードウェア空間の最適化の効率は高い。しかしながら、1つのハードウェアモジュールに対する用意されるアーキテクチャコードの量が多くなる。マッピングする際は、本発明においては、ハードウェアスペースさえあれば複数のオブジェクト回路をまとめてマッピングすることが可能である。したがって、オブジ

ェクト回路のサイズによりマッピングを繰り返す数が膨大になり、その結果、処理時間が増大するような心配は少ない。再構成可能なアーキテクチャの内、特定の数の再構成可能な回路により構成される回路ブロックを複数備えているアーキテクチャに対しては、アーキテクチャコードは、回路ブロックの単位でマッピングできるオブジェクト回路情報を含むことが望ましい。

[0033] 本発明のアーキテクチャコードは、回路を動的に再構成可能な論理回路領域を有するすべてのデータ処理装置に対して適用できる。しかしながら、FPGAのように回路の構成をルックアップテーブル(LUT)に記憶するハードウェアであると、LUTを変更するために数クロックを有し、実行速度の遅れが目立つ可能性がある。したがって、本発明においては、再構成に要する時間が非常に短い複数のエレメントを備えた論理回路領域を有するデータ処理装置を提供する。

[0034] 本発明のエレメントは、入力データを論理演算して出力データを出力する演算コアを備えており、演算コアは、論理演算を指示する多ビットのファンクションコードが入力され、入力データにより出力データを選択するセレクタを備えている。演算コアに入力されているファンクションコードを供給するだけで論理を変更できるので、LUTを書き換える必要がなく高速に論理を交換できる。

[0035] さらに、エレメントは、nを複数の整数としたときに、n個の入力と、n個の出力と、n個の入力から任意の入力データを選択する入力インタフェースと、n系統の入力および出力データのいずれかから任意に選択してn系統の出力の少なくともいずれかから出力する出力インタフェースとを備えていることが望ましい。例えば、2次元の広がりの論理回路領域であれば、複数nの候補は4であり、東西南北(前後左右)4方向のいずれからもデータを入力でき、4方向のいずれに対してもデータを出力できる。また、3次元の広がりのある論理回路領域であれば、複数nの候補は6であり、前後左右上下の6方向のいずれからもデータを入力でき、6方向のいずれに対してもデータを出力できる。さらに、このエレメントは論理演算しない単なる接続切り換えエレメントとしても機能する。さらに、演算コアに、n個の入力のいずれかのデータ、または、出力データをラッチするレジスタを設けることが望ましい。レジスタを使用しなければ、デコーダなどのクロック依存性のない、あるいは少ない処理を実行するのに適した回路を

構成でき、レジスタを使用すれば、ステートマシンなどのクロック依存性の高い処理に適した回路を構成できる。

[0036] 本発明によれば、数クロックあるいはサイクルの単位で実ハードウェア空間の動的な最適化が可能となる。このため、トレード・オフの自由度が非常に大きく、高性能高機能・低チップコスト・低消費電力という、相反する幾つかの要求を同時に高いレベルで実現することができる。したがって、リコンフィギャブル・テクノロジーの有効性は飛躍的に増大する。動的に再構成可能な回路領域における実装効率が向上し、専用LSIと比較して内部の稼働効率を格段に向上させ、チップ・コスト、性能及び消費電力の3つの点で有利な解決手段を提供することができる。また、動的に再構成可能なハードウェアがプログラマブルなハードウェアである特徴は最大限に活かされるので、従来のLSI開発手法では、物理デザインと機能検証・品質保証のために数ヶ月以上の時間を必要としたハードウェアの提供を、原理的にこれらの期間を必要としないアーキテクチャを提供できる。

[0037] すなわち、本発明においては、実現可能なハードウェア空間の動的最適化技術を提供している。従来、有体物で提供されたハードウェア回路情報と、命令プログラムのようなハードウェア回路を制御するソフトウェア情報とを1つの統一されたアーキテクチャコードと呼ばれる体系で符号化した新たな情報として提供している。このため、ハードウェアの実行サイクルにおいて、アプリケーション要求(アーキテクチャ)の拘束条件下で、その瞬間瞬間に有効となるハードウェア・リソースと、要求される処理性能とをトレードオフし、動的に最適化を行うことで、特定のアプリケーション用に設計された専用LSI対しても、リコンフィギャブルなチップが、チップ・コスト、性能要求・消費電力の3つの面で優位性を示すことが可能となる。

[0038] ハードウェア空間の動的最適化を実現するためのアーキテクチャの基本要素技術は、ハードウェア回路の時分割実行可能とする回路分割実行と継続実行技術、複数のハードウェア回路間のチャネル接続技術、動的ハードウェア回路生成技術・縮小技術・消去交換技術、ハードウェア回路情報のコンパクト化技術と回路情報の高速移動技術、アプリケーション要求とリソース間トレード・オフ・ソフトウェア技術、ハードウェア回路やソフトウェア情報の高速エミュレーション技術を挙げることができる。本発明

のアーキテクチャコードは、これらの技術をサポートできるものである。

[0039] また、本発明のアーキテクチャコードは、オブジェクト回路情報(分割回路情報)、インタフェース回路情報、境界条件を含むものであるが、さらには、アーキテクチャコードは、ハードウェア回路情報とソフトウェア情報の2つに大きく分類される。ハードウェア回路情報は、使用可能なハードウェア資源を100%とした場合の各回路のモジュール相関情報(静的トポロジー情報と動的モジュール実行情報)・階層構造・優先順位・例外処理条件・動的トレード・オフ条件等を含むことができる。アーキテクチャコードは、境界条件を始めとする、ハードウェア回路の機能やタイミング制御を補助的にサポートする情報全体を含むことができ、それには、従来の命令プログラムやベクターテーブル、アプリケーションによっては画像等のデータ情報といったソフトウェア情報も含まれる。

[0040] たとえば、マイクロ・プロセッサーの場合、外部割り込み制御部や例外処理部のような特定条件でしか必要とされない回路と、デコーダあるいはデータ・パスのように比較的高い頻度で実行されるような回路とが存在する。本発明のアーキテクチャコードを用いれば、ハードウェア空間に階層的に構成されるハードウェア回路を、マイクロ・プロセッサーという機能面から分析し直して、動的実行を想定した動的最適化が可能なように再構成した情報の集合体として、マイクロ・プロセッサーのハードウェアおよびソフトウェア情報を提供することができる。これは、一般的なプログラム空間とは異なり、並列実行を想定した複数のアーキテクチャコードから構成される。そして、アーキテクチャコードにより実装されるオブジェクト回路の内、回路交換の余裕が無いものに関するアーキテクチャコードはLSI内部のメモリに格納される。逆に、回路実行や起動余裕のあるオブジェクト回路に関するアーキテクチャコードは、外部メモリに格納され、外部メモリから内部メモリにロードされてから実行される。

[0041] 本発明のデータ処理装置であるアーキテクチャLSIは、ロードユニットおよびマッピングユニットとしての機能を備え、アーキテクチャコードをハードウェア上で翻訳してハードウェアの初期化や分割実行する為の外部・内部の高速ローディング制御ユニット(RLC)、実行制御ユニットとしての機能を備え、高速論理回路交換動作の制御と階層的情報伝達(伝送)を行う高速論理通信マスタ(RTM)、論理回路領域となる、

各種ハードウェア回路(テスト回路含む)を直接分割実行する高速論理回路交換エレメント(RXE)群から構成することができる。本発明の実施の形態については、本発明の他の課題、構成および効果と共に以下でさらに詳しく説明する。

図面の簡単な説明

[0042] [図1]本発明のデータ処理装置の概略構成を示す図である。

[図2]本発明のデータ処理装置の異なる例を示す図である。

[図3]アーキテクチャコードの概要を示す図である。

[図4]データ処理装置により制御されるロボットの概略構成を示す図である。

[図5]データ処理装置を備えた端末の概略構成を示す図である。

[図6]アーキテクチャコードを生成する過程を示すフローチャートである。

[図7]データ処理装置においてアーキテクチャコードを実行する過程を示すフローチャートである。

[図8]RC領域の構成の一例を示す図である。

[図9]RC領域の構成の異なる例を示す図である。

[図10]RC領域のハードウェア構成を示す図である。

[図11]エレメントの配置を示す図である。

[図12]エレメントの構成を示す図である。

[図13]演算コアの構成を示す図である。

[図14]演算コアの動作例を示す図である。

[図15]演算コアの他の動作例を示す図である。

[図16]演算コアで実行可能な論理演算の例を示す図である。

発明を実施するための最良の形態

[0043] 図1に、本発明のデータ処理装置の一例を示してある。このデータ処理装置1はアーキテクチャLSIであり、回路を動的に再構成可能な論理回路領域(RC領域、リコンフィグラブル領域)10と、幾つかのハードウェアモジュールのアーキテクチャコード20を記録したアーキテクチャライブラリ2と、アーキテクチャコード20を再構成可能なハードウェア10の上で翻訳してハードウェアの初期化や分割実行する高速ローディング制御ユニット(RLC)11と、高速論理回路交換動作の制御と階層的情報伝達(伝

送)を行う高速論理通信マスタ(RTM)12とを備えている。RLC11は、ライブラリ2からアーキテクチャコード20を取得(フェッチあるいはダウンロード)するロードユニット(LU)13としての機能を備えている。さらに、RLC11は、アーキテクチャコード20のオブジェクト回路情報およびインタフェース回路情報により、RC領域10にオブジェクト回路19と、そのオブジェクト回路19に接するインタフェース回路18とをマッピングするマッピングユニット(MU)14としての機能を備えている。RTM12は、アーキテクチャコードの境界条件にしたがってインタフェース回路18を制御する動作制御ユニットとしての機能を備えている。また、RTM12は、RC領域10に現在および/または過去にマッピングされたオブジェクト回路19のインタフェース回路18の状態を、必要に応じて境界情報メモリ15に記憶し、オブジェクト回路19の間の情報伝達を行う。

[0044] 以下の例においては、アーキテクチャコード20のオブジェクト回路情報は、ハードウェアモジュールを適当な範囲に分割して、RC領域10にマッピングできるように生成されている。したがって、オブジェクト回路情報によりRC領域10に構成されるオブジェクト回路19は、ある機能を実装するための単位として設計されたハードウェアモジュールを分割した分割回路となる。このため、以下においては、分割回路情報をオブジェクト回路情報として含むアーキテクチャコード20と、分割回路19とを例にして本発明をさらに説明する。

[0045] データ処理装置1は、さらに、アプリケーションを実行するプログラム4が記憶されたRAM5と、プログラム4にしたがってデータ処理装置1のハードウェア資源を用いて処理を実行するRISCプロセッサ6と、割り込み信号を受信する割り込み制御ユニット7と、データ処理装置1の各ハードウェア資源にクロック信号を供給するクロック発生源8と、外部メモリに対してデータの入出力を制御するデータ入出力インタフェース9とを備えている。コードRAM5は、RC領域10からもアクセスできるようになっている。

[0046] 図2に、本発明に係るデータ処理装置1の異なる例を示してある。CPUあるいはDSPとしてのハードウェアは、1つまたは複数のハードウェアモジュール(ハードウェアマクロ)により提供できる。同様に、アーキテクチャコード20をロードするロードユニット(LU)13、マッピングユニット(MU)14および動作制御ユニット(RTM)12の各機能も1つまたは複数のハードウェアモジュールにより提供できる。さらに、割り込み検出ユ

ニット(IU)7もハードウェアモジュールにより提供できる。したがって、これらのハードウェアモジュールを分割回路化して、アーキテクチャコードによりRC領域10に動的に構成することが可能である。このため、図2に示したデータ処理装置1は、RC領域10に、分割回路19とインタフェース回路18とによりプロセッサ6、LU13、MU14およびRTM12の機能が、部分的に、あるいは必要であれば全体として実現されるようになっている。

[0047] したがって、このデータ処理装置1においては、RISCプロセッサ6の機能がRC領域10を用いて実現されるので、RC領域10にマッピングされる回路によりRC領域10の制御も含めたデータ処理装置1の制御が行われる。この場合、データ処理装置1において構成が固定されたハードウェアで実現される機能は、データ処理装置であるアーキテクチャLSI1を起動する際、あるいはリセットする際に、そのための構成をRC領域10にマッピングする初期設定機能16となる。この初期設定機能16は、独立した回路であっても良いし、あるいは、RTM12のようにRC領域10を管理あるいは制御するために固定的に必要な機能に含まれて、その機能が固定されたハードウェアで提供されても良い。

[0048] このように、RC領域10にマッピングされる回路により、RC領域10の制御も含めたデータ処理装置1の制御を行うことができる。したがって、RAM4にデータ処理装置1の制御も含めたアーキテクチャコード28を用意することにより、アーキテクチャコード28によりデータ処理装置1の制御を行うことができる。このため、複数の命令セットを備えた従来のプログラムによる制御を、アーキテクチャコード28による制御に代えることが可能となる。このことは、記録媒体、ネットワーク、通信などによりアーキテクチャコード28を提供することにより、ハードウェア情報のみならず、従来のソフトウェア情報も含めてデータ処理装置1に実装できることを意味しており、データ処理装置1のフレキシビリティーは拡大する。

[0049] また、LU13、MU14およびRTM12の機能などのように、データ処理装置1において、アプリケーションを実行するために必要であるが、複数のアプリケーションに対して汎用性のある機能を提供するアーキテクチャコードは、データ処理装置1の基本的な機能をサポートするためのアーキテクチャコード29としてアプリケーション用のア

ーキテクチャコード20とは独立して供給することが可能である。このシステムをサポートするアーキテクチャコード29は、従来のプログラムベースで稼動するプロセッサにおけるOSあるいはBIOSに対応した取り扱いが可能となる。さらに、システムをサポートするハードウェアもアーキテクチャコード29として提供することにより、RC領域10の交換動作が不要な場合、あるいは交換動作のサイクルを低下できる場合は、RC領域10の交換動作のためのハードウェアが占める領域をアプリケーションの実行のためのハードウェアに解放することが可能となる。このため、RC領域10の利用効率を向上でき、少ないハードウェアリソースで最大限の処理能力を発揮するLSIを提供することが可能となる。

[0050] 図3にアーキテクチャコード20の一例を示してある。アーキテクチャコード20は、ハードウェア回路情報21とソフトウェア情報22から構成される。ハードウェア回路情報21は、ある機能単位を回路として実装するために設計されるハードウェアモジュール(IPまたはライブラリ)を複数に分割した分割回路19をRC領域10の一部にマッピングする分割回路情報23と、分割回路19に接するインタフェース回路18をRC領域10にマッピングするインタフェース回路情報24とを備えている。ソフトウェア情報22は、アーキテクチャコード20を識別するための識別情報25と、インタフェース回路18において実現する境界条件26に加え、その他の情報27、たとえば、他の分割回路に対する優先順位、例外処理条件、動的トレード・オフ条件、分割回路の実行順序などの情報を含む。アーキテクチャコード20により、回路を構成するプリミティブのファンクション情報やトポロジーはすべて明確にされ、個々の分割回路19のファンクションおよび分割回路19の接続関係も明確になる。

[0051] 本例のデータ処理装置1においては、たとえば、図1および図2に示したAという機能を実現するための回路構成(ハードウェアモジュール)は、A1ーAnのアーキテクチャコード20として与えられる。また、Bという機能を実現する回路構成は、B1ーBmのアーキテクチャコード20として与えられ、Cという機能を実現する回路構成は、C1ーCxのアーキテクチャコード20として与えられ、Dという機能を実現する回路構成は、D1ーDyのアーキテクチャコードとして与えられる。なお、n、m、xおよびyは適当な整数である。

[0052] 図4に示すように、データ処理装置1が、自動制御装置の一つの例であるロボット70の制御を行う場合、A機能は聴覚71の制御およびデータ処理を行い、B機能は視覚72の制御およびデータ処理を行い、C機能は言語能力73の制御およびデータ処理を行い、D機能は身体機能74の制御およびデータ処理を行う。そして、データ処理装置1のRC領域10には、これら複数の自動制御機構71ー74のハードウェアモジュール(AモジュールからDモジュール)の分割回路がそれぞれ動的にマッピングされる。RC領域10において、それぞれの機構71ー74のためのハードウェアモジュールが占める面積は動的に制御される。たとえば、ロボット70が会話するときは、そのために聴覚、視覚および言語能力が大きく要求されるので、機能A、BおよびCを制御するための分割回路19が占める面積が増大する。一方、ロボット70が身体を用いた作業を行うときは、視覚および身体機能の能力が大きく要求されるので、機能BおよびCを制御するための分割回路19が占める面積が増大する。

[0053] このロボット70は、さらに、外界、たとえば、無線あるいは有線を介してインターネットなどのコンピュータネットワークに接続するための機構75を備えている。したがって、それぞれの機能を実現するためのアーキテクチャコードを外界からダウンロードすることが可能となる。このため、ロボット70で実現できる作業の種類は基本的には限界がなくなる。そして、アーキテクチャコードの供給元において、アーキテクチャコードをアップデートすることにより、常に最新のハードウェアモジュールによりロボット70を制御することができるので、ハードウェアの陳腐化という問題も解決される。また、アーキテクチャコードを用いてロボット70を遠隔操作することも可能となる。外界との通信に必要な機能の内、回路でカバーできる機能は基本的にアーキテクチャコードを用いて本例のデータ処理装置1により行われる。したがって、通信用の機構75は、アンテナ、コネクタなどの、回路ではカバーできないハードウェアを備えたものとなる。

[0054] 1つのデータ処理装置1により通信までカバーできない場合は、通信用のデータ処理装置1を搭載し、通信機能をサポートすると共に、通信機能の負荷が小さいときは、他の機能をサポートするように設計することが可能である。他の機能についても同様であり、身体機能用として、1または複数のデータ処理装置1を集中あるいは分散して搭載し、身体機能の負荷が小さいときは、それらのデータ処理装置1を言語処理

などの他の機能ために用いることができる。

[0055] 図5に、データ処理装置1を搭載した端末80の概略構成を示してある。この端末80も外界と通信するための機構85を備えている。したがって、アーキテクチャコードを外界からダウンロードすることにより、端末80により様々なサービスを提供できる。ディスプレイ、マイク、スピーカなどの入出力機構81を搭載していれば、画像、音声などの入出力を必要とする全てのサービスを提供することができる。さらに、光、温度などを感知するセンサー82を備えていれば、カメラや温度計などとしてのサービスに限らず、身体をモニタするサービスなど、センサー82を用いた他の様々なサービスを提供することができる。そして、アーキテクチャコードを外界からダウンロードすることにより、常に最新のハードウェアモジュールの機能を利用できる。

[0056] 図6に、アーキテクチャコード20の生成方法を示してある。まず、ステップ31において、ハードウェアモジュールのオリジナルのネットリストを生成する。ネットリストを生成するまでの段階は、C言語などの高級言語、Verilogなどのハードウェア記述言語を用いた様々な方法が公知であり、いずれの方法を用いても良い。ステップ32において、オリジナルのネットリストを幾つかの単位に分割し、それぞれの分割した範囲内で、RC領域10にマッピングできるように配置および配線問題を解決して分割回路情報23を生成する。

[0057] RC領域10にマッピングする分割回路19は、RC領域10を構成するハードウェアの適切な範囲を単位とする回路ブロックを1または複数用いて配置されるように分割される。この分割方法を採用することにより、RC領域10に分割回路19を効率的に割付できる。また、分割回路19の間で頻繁にデータ交換が起きるような分割方法は、分割回路19を個別にRC領域10にマッピングすることを考えると好ましい分割方法とは言えない。もっとも、そのような分割回路19の組み合わせは、RC領域10の状況が許す限り同時にマッピングするように優先順位をつけることが本発明では可能である。したがって、本発明のデータ処理装置1においては、ネットリストの様々な分割方法を許容し、対応することが可能である。しかしながら、ステップ32においては、オリジナルのネットリストを分割し、それをRC領域10のハードウェアに割り当てる作業をある程度繰り返して最適な分割回路情報23が得られるようにすることが望ましい。

[0058] さらに、ステップ33において、オリジナルのネットリストの分割回路19の境界を形成する情報からインタフェース回路情報24を生成する。したがって、隣接する分割回路19と境界が一致する部分においては、インタフェース回路情報24は同一になり、部分的に共通のインタフェース回路情報24を持ったアーキテクチャコードが生成されることになる。

[0059] 次に、ステップ34において、オリジナルのネットリストを分割回路19の集合に変換し、それらの分割回路19の間で、ハードウェアモジュールとしての機能が実現されるように、タイミング収束問題などを含む配置および配線問題を解決し、インタフェース回路18における境界条件26を生成する。したがって、隣接する分割回路19と境界が一致または接続できる状態に対応しており、インタフェース回路情報24が同一または対応した構成となる部分においては、境界条件も同一または対応した条件になる。このため、部分的に共通の、あるいは対応した境界条件26を持ったアーキテクチャコードが生成されることになる。

[0060] さらに、分割回路を実ハードウェア空間(論理回路領域)10にマッピングしてハードウェアモジュールとしての機能を実現させるように上記の情報をコンパイルする段階で、他の分割回路に対する優先順位、例外処理条件、動的トレード・オフ条件、分割回路の実行順序などの情報が得られるので、それらを含め、ステップ35でアーキテクチャコード20を生成する。したがって、ハードウェアモジュールは複数の分割回路19により仮想ハードウェア空間上に構成され、アーキテクチャコード20により、その一部を実ハードウェア空間であるRC領域10に実現し、実行することが可能となる。RC領域10にマッピングされた分割回路19は、回路インスタンスであるということができる。

[0061] 仮想ハードウェア空間と実ハードウェア空間とはインタフェース回路19を介して結び付けられており、タイミング収束などの実ハードウェア空間に配置配線する際の問題は、インタフェース回路19を境界条件26により制御するという解決策が示されている。したがって、仮想ハードウェア空間においても、実ハードウェア空間においても、所望の分割回路19を用いてソフトウェア的な処理、およびハードウェア的な処理を実現することが可能となる。

[0062] 図7に、データ処理装置1において、アーキテクチャコード20を用いてRC領域10

に分割回路19およびインタフェース回路18を生成して実行する過程を示してある。まず、ステップ41において、ロードユニット13は、RTM12により指定されたアーキテクチャコード20をライブラリ2から取得する。本例のデータ処理装置1において、リスクプロセッサ6がアプリケーションプログラム4を実行するケース、アーキテクチャコード28によりRC領域10に実現される構成によりデータ処理装置が制御されるケースがあるが、いずれにおいても、RTM12が動作環境情報に基づき、取得するアーキテクチャコードを決定し、ロードユニット13に指示を出す。動作環境情報は、プログラム4あるいはアーキテクチャコード28により得られる当該データ処理装置1に対する要求、割り込み制御回路7からの割り込み情報、RC領域10にマッピングされた分割回路19の実行状況、RC領域10の利用可能状況(空き領域の有無、置換可能な分割回路の有無など)が含まれる。

[0063] ロードユニット13は、ライブラリ2にコード20があればアドレスを出力してコード20をフェッチする。ロードユニット13が適当な通信機能を備えていれば、他のデータ処理装置や、外部のメモリ、さらには、ネットワークで接続されたサーバやその他のネットワーク上の資源からコード20を取得することができる。アーキテクチャコード28のように、アーキテクチャコードが強制的に、あるいは能動的にロードユニット13にロードされる構成にすることにより、アーキテクチャコードを介してデータ処理装置1における処理を能動的に制御することも可能である。

[0064] ステップ42において、マッピングユニット14は取得されたアーキテクチャコード20の分割回路情報23およびインタフェース回路情報24により、RC領域10に分割回路19と、その分割回路に接するインタフェース回路18とをマッピングする。RC領域10の状況は、分割回路19の実行を制御するRTM12が最も精度良く把握できるので、マッピングユニット14はRTM12の指示により、RC領域10の空いたハードウェア空間またはリプレイス可能なハードウェア空間に分割回路19およびインタフェース回路18をマッピングする。その際、隣り合う分割回路19との境界のインタフェース回路情報24および境界条件26が一致あるいは対応する場合は、仮想ハードウェア空間において隣接する分割回路19なので、実ハードウェア空間10においてそのまま接続することが可能である。したがって、双方のインタフェース回路を経ずに隣り合う分割回路同

士が接続されるように分割回路19をマッピングする。なお、図1および図2などにおいて、表示を簡略にするためにインタフェース回路18は分割回路19の左右のみに形成されている。分割回路19が左右のみならず上下にも接続する配線を含む場合は、インタフェース回路18は分割回路19の上下左右に分割回路19をカプセルで包むようにアレンジされる。さらに、RC領域10が3次元方向の広がりを持ち、分割回路19も3次元方向の広がりを持つ回路であれば、インタフェース回路18は、立体的に分割回路19をエンカプセルするように構成される。

[0065] 基本的には、RC領域10の空き領域に分割回路19はマッピングされることになる。RTM12が把握している動作環境情報によると、RC領域10にマッピング済みの他の分割回路に対して、新たな分割回路19を優先してマッピングする緊急性があれば、マッピング済みの他の分割回路19を消去したり、縮小して空き領域を形成することも可能である。消去した他の分割回路19は、緊急性が除かれた後に、RC領域10に再度マッピングして、最初から、あるいは途中から実行することができる。また、縮小した他の分割回路19では、分割回路19をマッピングする工程を繰り返すことにより、処理速度は低下するが、その分割回路に係る機能の処理を継続して実行することができる。このように、本例のデータ処理装置1においては、RC領域10に、動作環境情報などには依存するが、神出鬼没に分割回路19をマッピングすることができる。アプリケーションが直面する場面を予めシミュレーションなどにより把握し、RC領域10の所定の位置に所望の分割回路19がマッピングされるようにスケジューリングすることも可能であり、RC領域10の利用効率を向上することができる。

[0066] ステップ43において、マッピングされた分割回路19を動作させる。分割回路19を動作させるために、ステップ44において、境界条件26に基づきインタフェース回路18を制御し、分割回路19に所定のタイミングで所定のデータを供給する。このステップ44において、RTM12の機能により、RC領域10に現在および／または過去に、時間的および／または空間的に分かれてマッピングされた他の分割回路19のインタフェース回路18の状態が、境界条件26に基づき、動作対象の分割回路19のインタフェース回路18の制御に反映される。したがって、ステップ45において、実ハードウェア空間に実現された分割回路19は、周囲に他の分割回路が接続されている仮想

ハードウェア空間と同じ状態となり、その分割回路19が属するハードウェアモジュールとしての機能が実ハードウェア空間上で実現される。また、分割回路19が動作した結果は、インタフェース回路18に出力されるので、RTM12はそのインタフェース回路18の状態をRC領域10にマッピングされている他の分割回路19のインタフェース回路18に空間的に伝達したり、次にマッピングされる他の分割回路19のインタフェース回路18に時間を経て伝達する。これにより仮想ハードウェア空間では信号がネットリストに従って伝播し、ハードウェアモジュールの機能が実現される。

[0067] インタフェース回路18に設定する境界情報は、メモリ15に記憶しておくことが可能である。他の分割回路19がマッピングされるタイミングまでの時間が長かったり、動作途中に分割回路19が消去されたときに、メモリ15に記憶された境界情報をインタフェース回路18に設定することにより、分割回路19を所望の条件で動作、あるいは再動作させることができる。

[0068] ステップ46において、マッピングされた分割回路19を動作させる要求が終了するまでステップ44および45を繰り返す。そして、処理が終了した分割回路19は、ステップ47においてRC領域10から消去される。あるいは、RC領域10に余裕があり、以降に分割回路19の機能が必要となることが予測される場合は、縮小してRC領域10に存在させることも可能である。さらに、RC領域10に余裕がある場合は、そのまま存在させておいても良い。

[0069] 連続して入力されるデータに対して繰り返し動作が必要な分割回路19がマッピングされている場合は、その処理が終了するまで同一の分割回路19がRC領域10に存在する。並列度を高めることにより処理速度が上がる場合は、同一の分割回路19の回路インスタンスを複数マッピングして処理速度を向上することも可能である。さらに、同一のアーキテクチャコード20に対して複数の回路インスタンスをRC領域10に構成し、それらの出力を比較することにより信頼性の高い処理を実現することも可能である。そして、信頼性が要求される状態であることを判断したり、処理速度が要求される状態であることを判断して、そのような構成を自動的に採用するようにデータ処理装置1を制御することも可能である。一方、ステートマシンのように、ステートが進むことにより処理内容が順番に変わる場合は、次々に異なる分割回路19がマッピングさ

れる。

[0070] 分割回路19、すなわち、回路インスタンスは、マッピングされる際に、ハードウェア空間の動的最適化を行う為に、他の回路インスタンスの起動と消去をテーブルマスタであるRTM12に要求することができる。RTM12は、複数の回路生成・消去・コピー・移動や回路間のチャネル接続を行い、本来は大規模な回路を物理空間上に展開し回路構成しないと動作しない機能を、瞬間瞬間で必要な回路だけを回路インスタンスとしてハードウェア空間に動的に最適化しながら生成し、資源の少ないハードウェア空間を用いて実質的には膨大な複数の回路を並列に動作させることができる。

[0071] ハードウェア空間に生成される分割回路19は、常に、このデータ処理装置(アーキテクチャLSI)1の論理回路領域(回路プレーン)10の上に存在するパーマネント回路と、生成された回路がある一定時間しか存在しないインスタント回路、一定時間毎に生成されるサイクリック回路等の種類に分けることができる。インスタント回路やサイクリック回路は、実際に実行されると消去される前に自分の実行結果で他の回路に通知すべき情報をRTM12へ通知し記憶させておく。通常は、この回路実行情報は、次に生成される分割回路19へ効率良く伝達される。逆に、RTM12は、インスタント回路間の実行情報が効率良く伝達されるように回路制御を行う。

[0072] 分割回路19の実行順序の確定は、図6に示したアーキテクチャコード20を生成する開発段階で、開発環境(FW)の回路コンパイラがこれを行う。分割回路が、外部信号やデータ入力条件により回路実行順序に変更がある場合は、RTM12が、この実行制御を行う。逆に、分割回路自身で実行順番が完全に制御可能な場合は、RTM12がシステム全体での優先順位に応じて回路の実行エリアの拡大・縮小を行う。

[0073] たとえば、図1のRC領域10には、A機能を実現するAモジュールの分割回路A1がインタフェース回路と共に生成され、B機能を実現するBモジュールの分割回路B1ーB3がインタフェース回路と共に生成されている。分割回路B1ーB3は、連続した回路インスタンスで連続したRC領域10に生成されたので、隣接した分割回路の境界領域は連続しており、連続した分割回路から外側に繋がる境界にインタフェース回路18が形成されている。なお、簡単に説明するためにインタフェース回路18が図面の左右のみに生成されているが、仮想ハードウェア空間において上下に分割回路が接続

される場合は、インタフェース回路が生成される場合があることは上述したとおりである。

[0074] C機能を実現するCモジュールにおいては、分割回路C1およびC2がRC領域10にマッピングされているが空間的に分割されている。このため、各々の分割回路C1およびC2にインタフェース回路18が生成され、RTM12を介してこれらの分割回路C1およびC2は接続される。また、D機能を実現するDモジュールにおいては、分割回路D1およびD2が接続した状態でマッピングされている。RTM12は、これらの分割回路19のインタフェース回路18に適当なタイミングでデータをセットすることにより分割回路18をアクティブにし、その結果インタフェース回路18に出力されたデータを保存したり、空間あるいは時間的に分割された接続先の分割回路19のインタフェース回路18に伝達する。

[0075] さらに、RTM12は、分割回路19のアーキテクチャコード20の情報や、分割回路19に対する動作環境情報により、RC領域10の分割回路19に対してクロック発生源8から供給されるクロック信号の種類、すなわち周波数を変えることができる。このため、RC領域10の電力消費を必要最小限に抑えることができ、パフォーマンスは最大に維持することができる。RC領域10のうち、回路インスタンスがマッピングされていない領域にはクロック信号は原則として供給されない。

[0076] 図8および図9は、時間が経過したRC領域10の状態である。A機能はインスタント回路であり、A1、A2およびA3という分割回路19が次々と生成されては消滅していき、その間のデータの転送はRTM12により行われる。B機能は、図示したシーケンスでは緊急性を要する機能としてRTM12に要求されており、RC領域10のかなりのハードウェア資源を費やして生成されている。図8に示したタイミングでは、D機能を消滅させ、その資源を用いて多数の分割回路19が生成されている。したがって、図9に示したタイミングでは、B機能の分割回路19が消滅した領域にD機能の分割回路19を復元して、再度、D機能の処理を途中または初めから再実行することになる。

[0077] 図10に、RC領域10の構成を示してある。本例のRC領域10は、各々の論理演算を変更可能な複数のエレメントの集合である回路ブロック(rxe__plane)51が格子状(アレイ状あるいはマトリクス状)に配列され、それらの間が配線52により接続されてい

る。アーキテクチャコード20により定義される分割回路19のサイズは、この回路ブロック51の倍数を単位とすることが望ましい。その場合、分割回路情報24をコンテキスト(コンフィグレーション情報)として、分割回路19が1つまたは複数の回路ブロック51を消費してマッピングされる。

[0078] 図11に、1つの回路ブロック51の構成を示してある。本例では、回路ブロック51には、16個の論理エレメント53が4×4のアレイ構造をなすように配列されている。各々の論理エレメント53は図面の上下左右の4方向に隣接する論理エレメント53と4ビットのレイア1のバス54により接続されている。さらに、上下左右に隣接する論理エレメント53を越して、その外側に位置する論理エレメント53と接続するレイア2のバス55も用意されている。このため、論理エレメント53の間を、よりフレキシブルに接続することができる。さらに、論理エレメント53を3つ飛び越したレイア3のバスを配置することも可能である。

[0079] 各々の論理エレメント53は、論理演算エレメントとしての機能と、論理エレメント間の接続切り換えを行う配線スイッチとしての機能を備えている。そして、演算する論理と、配線接続の状態を高速で変更または交換する必要があるので、本例のRC領域10には、RXE(Rapid eXchange Element)53と称される高速で交換動作が可能なエレメントが配置されている。

[0080] 図12に、RXE53の構成を示してある。RXE53は、4系統の入力61と、4系統の出力62と、4系統の入力61から任意の入力データを選択する入力インタフェース63と、この入力インタフェース63により選択された入力データφiを論理演算してデータを出力する演算コア65と、4系統の入力61と演算コア65の出力データφoとを任意に選択して4系統の出力62へ接続可能な出力インタフェース64とを備えている。演算コア65は、論理演算を変更可能な構成になっており、論理を変更可能な演算エレメントとしての機能を果たす。また、入力インタフェース63は、4系統の入力61から任意の1ビットを選択するための16対1のセレクタ63sが複数配置された構成となっている。出力インタフェース64は、演算コア65からの出力φoと4系統の入力61のルーティングを兼ねた7対1のセレクタ64sが複数配置された構成となっている。

[0081] 図13に、演算コア(rxe__core)65の構成を示してある。演算コア65は、論理演算

を指示する16ビットのファンクションコードφfを入力とし、入力データφiにより出力データφoを選択するセレクタ66を備えている。演算コア65は、さらに、4ビットの入力データφiをデコードして16ビットのセレクタ66の選択信号を生成するデコーダ67と、4系統の入力61のいずれかのデータ、または、出力データφoをラッチするレジスタ68と、レジスタ68にラッチする信号を選択するためのセレクタ69aおよび69bとを備えている。

[0082] 図14および図15に、演算コア65の動作を示している。演算コア65はモード信号φmによって動作が変わる。図14のモード0は、演算コア65は、4ビットの入力データφiにより1ビットの出力データφoを生成し、その出力データφoをレジスタ68でラッチして出力する。図14のモード1は、演算コア65は、4ビットの入力データφiにより1ビットの出力データφoを生成し、その出力データφoをレジスタ68でラッチせずに出力する。出力データφoは、16ビットのファンクションコードφfと、入力データφiをデコードした結果による。したがって、図16に示すように、これらのモード1および2においては、ファンクションコードφfを変えることにより、演算コア65を4入力ANDから4入力コンパレータまで、9種類以上の異なる論理演算素子として使用することができる。

[0083] さらに、演算コア65は、セレクタ66とファンクションコードφfの組み合わせに論理演算を行っている。このため、従来のFPGAのようにSRAMなどの記憶素子を用いたルックアップテーブル(LUT)に論理をセットする必要がない。したがって、SRAMに入出力を行うサイクルを省略することができ、ファンクションコードφfを演算コア65に出力したタイミングで瞬時に演算コア65で行う演算を交換することができる。このため、本例の演算コア65は高速交換演算素子と称されている。

[0084] 図15に示したモード2からモード4においては、1つの演算コア65が、2ビットの入力信号φiに対して1ビットの出力信号φoを出力する2つの演算素子として機能する。すなわち、内蔵された16対1のセレクタ66が、2つの4対1のセレクタとして動作するようにセットされる。これらのモード2から4においては、演算コア65は、図16に示してあるように、ファンクションコードφfを変えることにより、インバータから2入力EXNORまで、7種類以上の異なる論理演算素子として使用することができる。

[0085] さらに、図15に示したモード5からモード7においては、演算コア65を、3ビットの入

力信号φiに対して1ビットの出力信号φoを出力する演算素子として使用できる。追加ビットの入力を許せば、内蔵された16対1のセレクタ66を、2つの3対1のセレクタとして動作するようにセットできるので、演算コア65を2つの3ビット入力1ビット出力の演算素子としても利用できる。これらのモード5から7においては、演算コア65は、図16に示してあるように、ファンクションコードφfを変えることにより、3入力ANDからフルアダーまで、5種類以上の異なる論理演算素子として使用することができる。

[0086] このように、本例のRC領域10を構成するRXE53は、セレクタ方式で高速で論理を交換することが可能である。RXE53は、さらに、内部に出力データをラッチするレジスタ68を備えており、出力データをスルーで出力することも、F／Fによりクロックに同期した状態でも出力することができる。したがって、デジタル回路で良く使用される組み合わせ回路(デコーダ)と、順序回路(ステートマシン)及び演算回路(データパス)を、アーキテクチャコード20の回路情報により効率よく実装し、実行することができる。

[0087] 本例の論理を再構成可能なエレメント(RXE)53は、2次元アレイあるいはマトリクスを構成すること考えている。したがって、2次元に格子状に配置するのに適した4系統の入出力を備えている。しかしながら、エレメント間を接続するネットワークが1次元的であれば、2系統あるいは3系統の入出力で対応することができる。さらには、エレメント間を接続するネットワークが3次元的であれば、5系統以上の入出力を用意することが望ましい。さらに、本例の演算コア(rxe__core)は、高速で交換動作が可能なようにセレクタ方式を採用しているが、ルックアップテーブル(LUT)へロジックを入力するサイクルを消費できるようであればLUTを備えた演算コアを採用することも可能である。

[0088] また、本例においては、同一構造のエレメント53によりマトリクスを構成しているが、論理演算用のエレメントとネットワーク形成用のエレメントによりマトリクスを構成することも可能である。さらに、算術計算を主としたエレメント、アドレス発生を主としたエレメントなどのある程度の機能に特化した、または、汎用性はあるが、ある機能の処理能力の高い複数種類のエレメントを適当な密度で配置したマトリクスにより、回路を再構成できるRC領域を構成することも可能である。

産業上の利用可能性

[0089] システムのハードウェア・アーキテクチャは、一般的に設計開始(検討)の段階で要求仕様として確定することが多い。実際のアプリケーションが固まった段階における要求の変化や、設計初期段階の時点では予想しなかった要求仕様の変更に対応するために、最近のFPGAやPLDは、ハードウェア構成を変更可能なアーキテクチャを採用している。しかしながら、その柔軟性自体は、内部を構成する基本エレメントを冗長化し、チップ・コストの競争力の点、および専用設計されたLSIやASSPに比較して動作周波数の点で不利な要素となる。

[0090] 最近は、ダイナミック・リコンフィギャブル・マシンが注目されるようになり、チップ・コストが高いという問題と動作周波数が低いという欠点をカバーできるようになりつつある。ただ、その競争力は、1ー2年掛けて開発された専用LSIと比較すると十分なレベルにはない。本発明においては、これらの問題を解決することに加え、低消費電力化も実現することで、トータルとして現在のSoCのコストパフォーマンスを実現しながら、アーキテクチャへの動的最適化を行い、次に来るハイパーSoCを実現することができる。

[0091] 一方、現在のLSI開発の問題は、チップ・コストの競争力という点と、性能・低消費電力化は最高であっても開発期間と開発コストは最悪という点であるが、これらの問題も解決することができる。

[0092] 現在のLSI設計の常識では、ハードウェア記述言語(Verilog-HDLやVHDL)を中心にして、これを各社の半導体プロセスに合うライブラリの接続形式に合うネットリストに翻訳(論理合成)する。この場合、物理配線と各論理ゲート(回路)の接続形態により動作周波数も影響を受けるが、それにも増して大きな問題は、システムアーキテクチャの視点からの最適化が出来ない点である。つまり、現在のSoCやFPGA、ダイナミック・リコンフィギャブル技術は、ハードウェアを実行する際に、アーキテクチャ・レベルからの動的最適化を実現できない構造になっている。本発明は、ハードウェアを実行する際のハードウェア空間を動的に最適化することが可能であり、この問題を解決する。

[0093] また、現在のLSI開発手法と実装方法では、システムの信頼性を上げたり品質を保

証したりするためのコストが異常に大きいと言える。1つの要因は、テスト回路を実装しないと内部の機能チェックができず、テスト回路を実装するとそのテスト回路でチップ面積が占められ、チップコストが上昇することである。したがって、結果的に品質を上げる手段は存在するが、最終的にはコストとのトレード・オフとなり、信頼性や品質保証をするには限界がある。このため、コンシュマー品に最も必要とされるテスト自体が製品の競争力を奪う結果となってしまう。さらに、デバッグを容易化するための設計も全体の開発時間や開発リソースを減少するために必要なコンセプトであるが、やはり、その設計のための費用がチップ・コストを上昇させる要因となる。

[0094] 本発明は、これらの課題の全てに対して解を与えることができる。本発明のハードウェア空間の動的最適化テクノロジーは、信頼性や品質保証する回路を必要なタイミングだけ存在させて、全体的なコスト影響を最小にできる。デバッグ容易化のための回路は、デバッグが完了すれば一般的には必要無い。逆に、デバッグが必要なタイミングで追加すべきデバッグ用回路を生成すれば良く、本発明においては、極めて容易に対応できる。

[0095] さらに、アーキテクチャコードに基づく本発明は、将来、ネットワーク等を使って、動的にテスト回路やその他の機能を実現する回路を変更したり、生成したりすることを可能とし、大規模で複雑なシステムを構築するコストを大幅に低減できる。したがって、手元には小型のチップ化された本発明のデータ処理装置が内蔵されたターミナルを持ち、ネットワークを介して膨大なリソースを持つ仮想ハードウェア空間と接続することにより、多種多様な機能を手元の小型のターミナルにより実行することが可能となる。このシステムは、ネットワークを介して膨大な入出力データを通信しながらネットワーク上に存在するハードウェア資源を用いて処理を行う現在の方式とは全く逆の発想であり、ネットワーク上に存在するハードウェア資源を手元のターミナルで実行しようというものである。したがって、大量の入出力データの送受信を緩和してネットワーク負荷を低減でき、また、データの秘匿性を保証できるなど、様々なメリットを持ったシステムが本発明に基づき構築されうる。

[0096] また、上記においては、半導体集積回路技術をベースにしたLSIに本発明を適用する例を説明しているが、いわゆる回路網を形成するデータ処理装置のすべてに本

発明を適用することが可能である。すなわち、電気あるいは電子レベルの回路技術をベースにしたデータ処理装置に限らず、光、生体、分子あるいは原子構造、遺伝子構造などをベースにした回路網を形成する全てのデータ処理装置に対して本発明を適用することができる。

請求の範囲

[1] 回路を動的に再構成可能な論理回路領域を有するデータ処理装置の制御方法であって、

アプリケーションを実行するための回路の少なくとも一部であるオブジェクト回路を前記論理回路領域の一部にマッピングするためのオブジェクト回路情報と、前記オブジェクト回路に接するインタフェース回路を前記論理回路領域にマッピングするためのインタフェース回路情報と、前記インタフェース回路において実現する境界条件とを備えたアーキテクチャコードを取得する工程と、

前記アーキテクチャコードの前記オブジェクト回路情報およびインタフェース回路情報により、前記論理回路領域に前記オブジェクト回路と、そのオブジェクト回路に接する前記インタフェース回路とをマッピングする工程と、

前記アーキテクチャコードの前記境界条件に基づき前記インタフェース回路を制御する動作工程とを有する制御方法。

[2] 前記オブジェクト回路は、ある機能を実装するためのハードウェアモジュールを分割した分割回路である、請求項1の制御方法。

[3] 前記マッピングする工程では、前記オブジェクト回路と前記インタフェース回路とを、前記論理回路領域の利用可能ないずれかの領域にマッピングする、請求項1の制御方法。

[4] 前記マッピングする工程では、隣り合うオブジェクト回路との境界におけるインタフェース回路情報および境界条件が一致または対応する場合は、双方のインタフェース回路を経ずに前記隣り合うオブジェクト回路と接続するように前記オブジェクト回路をマッピングする、請求項1の制御方法。

[5] 前記動作工程では、前記論理回路領域に、時間的および／または空間的に分かれてマッピングされた他のオブジェクト回路のインタフェース回路の状態が前記境界条件に基づき当該オブジェクト回路のインタフェース回路の制御に反映される、請求項1の制御方法。

[6] 前記取得する工程では、当該データ処理装置に対する要求、マッピングされた前記オブジェクト回路の実行状況、前記論理回路領域の利用可能状況を含めた動作

環境情報に基づき、取得する前記アーキテクチャコードを選択する、請求項1の制御方法。

[7] 前記取得する工程では、前記アーキテクチャコードを、通信ネットワークを介して取得する、請求項1の制御方法。

[8] 前記論理回路領域は、特定の数の再構成可能な複数のエレメントにより構成される回路ブロックを複数備えており、前記アーキテクチャコードは、前記回路ブロックの単位の前記オブジェクト回路情報を含む、請求項1の制御方法。

[9] 回路を動的に再構成可能な論理回路領域と、

アプリケーションを実行するための回路の少なくとも一部であるオブジェクト回路を前記論理回路領域の一部にマッピングするためのオブジェクト回路情報と、前記オブジェクト回路に接するインタフェース回路を前記論理回路領域にマッピングするためのインタフェース回路情報と、前記インタフェース回路において実現する境界条件とを備えたアーキテクチャコードを取得するロードユニットと、

前記アーキテクチャコードの前記オブジェクト回路情報およびインタフェース回路情報により、前記論理回路領域に前記オブジェクト回路と、そのオブジェクト回路に接する前記インタフェース回路とをマッピングするマッピングユニットと、

前記アーキテクチャコードの前記境界条件にしたがって前記インタフェース回路を制御する動作制御ユニットとを有するデータ処理装置。

[10] 前記オブジェクト回路は、ある機能を実装するためのハードウェアモジュールを分割した分割回路である、請求項9のデータ処理装置。

[11] 前記ロードユニット、前記マッピングユニットおよび前記動作制御ユニットは、それぞれ前記ハードウェアモジュールの1つであり、前記分割回路により前記論理回路領域に実装される、請求項10のデータ処理装置。

[12] 前記マッピングユニットは、前記オブジェクト回路と前記インタフェース回路とを、前記論理回路領域の利用可能ないずれかの領域にマッピングする、請求項9のデータ処理装置。

[13] 前記マッピングユニットは、隣り合うオブジェクト回路との境界における前記インタフェース回路情報および境界条件が一致または対応する場合は、双方のインタフェー

ス回路を経ずに前記隣り合うオブジェクト回路と接続するように前記オブジェクト回路をマッピングする、請求項9のデータ処理装置。

[14] 前記動作制御ユニットは、前記論理回路領域に、時間的および／または空間的に分かれてマッピングされた他のオブジェクト回路のインタフェース回路の状態を前記境界条件に基づき当該オブジェクト回路の前記インタフェース回路の制御に反映する、請求項9のデータ処理装置。

[15] 前記論理回路領域に、時間的および／または空間的に分かれてマッピングされたオブジェクト回路のインタフェース回路の状態を記憶する境界情報メモリをさらに有する、請求項9のデータ処理装置。

[16] 前記ロードユニットは、当該データ処理装置に対する要求、マッピングされた前記オブジェクト回路の実行状況、前記論理回路領域の利用可能状況を含めた動作環境情報に基づき、複数のアーキテクチャコードを備えたアーキテクチャライブラリから前記アーキテクチャコードを取得する、請求項9のデータ処理装置。

[17] 前記ロードユニットは、前記アーキテクチャコードを、通信ネットワークを介して取得する、請求項9のデータ処理装置。

[18] 複数のハードウェアモジュールを構成する複数のアーキテクチャコードを備えたアーキテクチャライブラリを有する、請求項10のデータ処理装置。

[19] 前記論理回路領域は、特定の数の再構成可能なエレメントにより構成される回路ブロックを複数備えており、前記アーキテクチャコードは、前記回路ブロックの単位の前記オブジェクト回路情報を含む、請求項9のデータ処理装置。

[20] 前記エレメントは、複数n個の入力と、n個の出力と、

前記n個の入力から任意の入力データを選択する入力インタフェースと、

この入力インタフェースにより選択された入力データを論理演算して出力データを出力する演算コアであって、その論理演算を変更可能な演算コアと、

前記n個の入力および前記出力データの少なくともいずれかを任意に選択して前記n個の出力の少なくともいずれかから出力する出力インタフェースとを備えている、請求項19のデータ処理装置。

[21] 前記演算コアは、論理演算を指示する多ビットのファンクションコードが入力され、

前記入力データにより前記出力データを選択するセレクタを備えている、請求項20のデータ処理装置。

[22] 前記演算コアは、前記n個の入力のいずれかのデータ、または、前記出力データをラッチするレジスタを備えている、請求項20のデータ処理装置。

[23] 請求項10に記載のデータ処理装置と、

前記データ処理装置に実装される少なくとも1つの前記ハードウェアモジュールが、それぞれの制御またはデータ処理に用いられる複数の自動制御機構とを有し、

前記論理回路領域に、複数のハードウェアモジュールの分割回路がそれぞれ動的にマッピングされる、自動制御装置。

[24] 前記アーキテクチャコードを外界との通信により取得するための通信機構をさらに有する、請求項23の自動制御装置。

[25] 請求項9に記載のデータ処理装置と、

前記アーキテクチャコードを外界との通信により取得する通信機構とを有する端末。

[26] 回路を動的に再構成可能な論理回路領域を有するデータ処理装置を制御するためのアーキテクチャコードであって、アプリケーションを実行するための回路の少なくとも一部であるオブジェクト回路を前記論理回路領域の一部にマッピングするためのオブジェクト回路情報と、前記オブジェクト回路に接するインタフェース回路を前記論理回路領域にマッピングするためのインタフェース回路情報と、前記インタフェース回路において実現する境界条件とを備えたアーキテクチャコードを記録していることを特徴とする記録媒体。

[27] 前記オブジェクト回路は、ある機能を実装するためのハードウェアモジュールを分割した分割回路である、請求項26の記録媒体。

[28] 回路を動的に再構成可能な論理回路領域を有するデータ処理装置を制御するためのアーキテクチャコードであって、アプリケーションを実行するための回路の一部であるオブジェクト回路を前記論理回路領域の一部にマッピングするためのオブジェクト回路情報と、前記オブジェクト回路に接するインタフェース回路を前記論理回路領域にマッピングするためのインタフェース回路情報と、前記インタフェース回路におい

て実現する境界条件とを備えたアーキテクチャコードの生成方法であって、

前記アプリケーションを実行するための回路のネットリストを分割し、それぞれの分割した範囲内の配置・配線問題を解決して前記オブジェクト回路情報を生成する工程と、

前記ネットリストの、前記オブジェクト回路情報により構成される前記オブジェクト回路の境界を形成する情報から前記インタフェース回路情報を生成する工程と、

前記ネットリストを前記オブジェクト回路の集合に変換し、それらのオブジェクト回路の間の配置・配線問題を解決し、それぞれの前記オブジェクト回路のインタフェース回路における境界条件を生成する工程とを有する、アーキテクチャコードの生成方法。

[29] 前記生成する工程では、ある機能を実装するためのハードウェアモジュールのネットリストを分割して前記オブジェクト回路情報を生成する、請求項28の生成方法。

[30] 回路を動的に再構成可能な論理回路領域を有するデータ処理装置であって、

前記論理回路領域は再構成可能な複数のエレメントを備え、

前記エレメントは、入力データを論理演算して出力データを出力する演算コアを備えており、前記演算コアは、論理演算を指示する多ビットのファンクションコードが入力され、前記入力データにより前記出力データを選択するセレクタを備えている、データ処理装置。

[31] 前記エレメントは、複数n個の入力と、n個の出力と、

前記n個の入力から前記入力データを選択する入力インタフェースと、

前記n個の入力および前記出力データの少なくともいずれを選択して前記n個の出力の少なくともいずれかから出力する出力インタフェースとを備えている、請求項30のデータ処理装置。

[32] 前記エレメントは、4系統の入力と、4系統の出力と、

前記4系統の入力から任意の前記入力データを選択する入力インタフェースと、

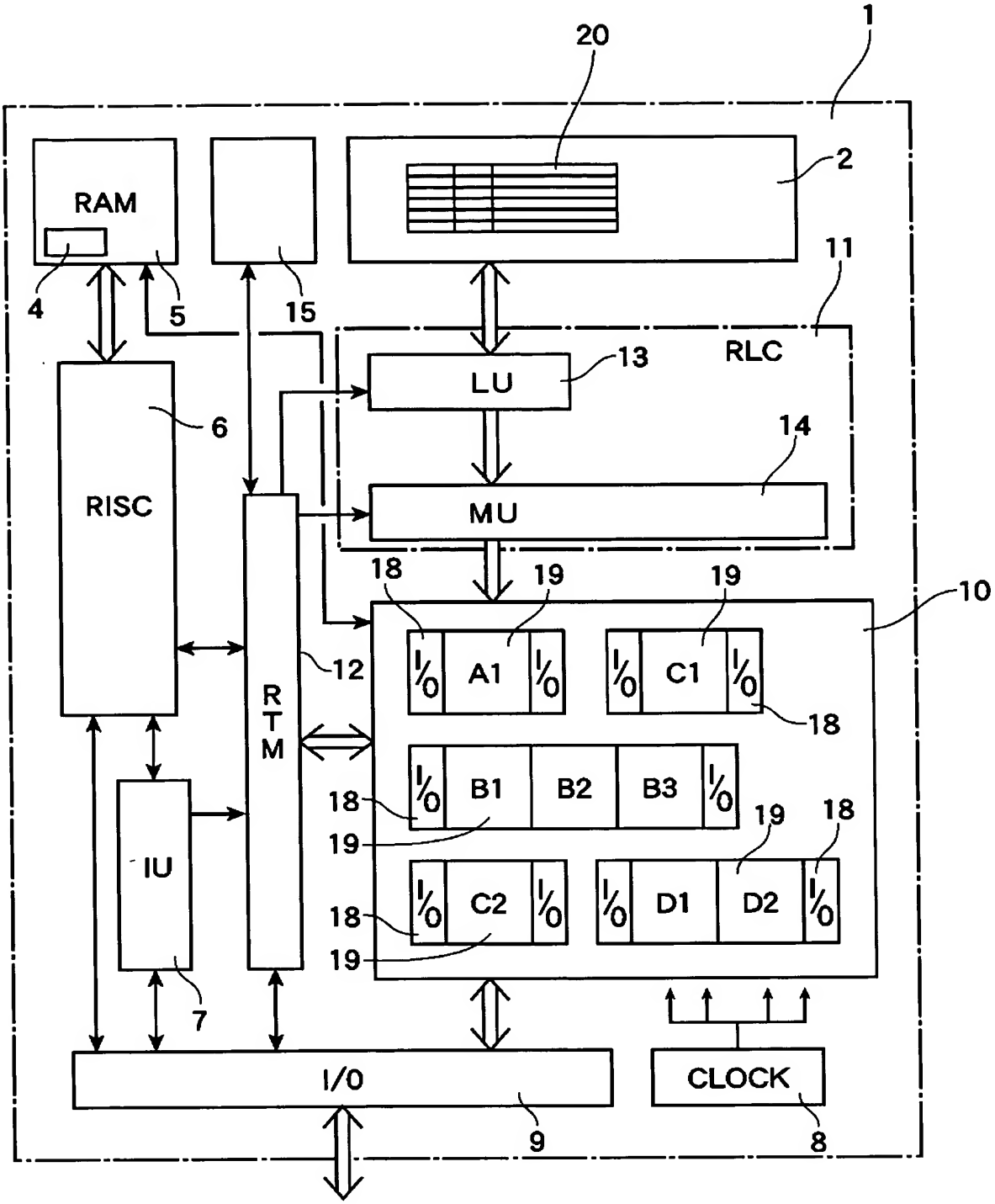
前記4系統の入力と前記出力データとを任意に選択して前記4系統の出力へ接続可能な出力インタフェースとを備えている、請求項30のデータ処理装置。

[33] 前記演算コアは、前記n個の入力のいずれかのデータ、または、前記出力データを

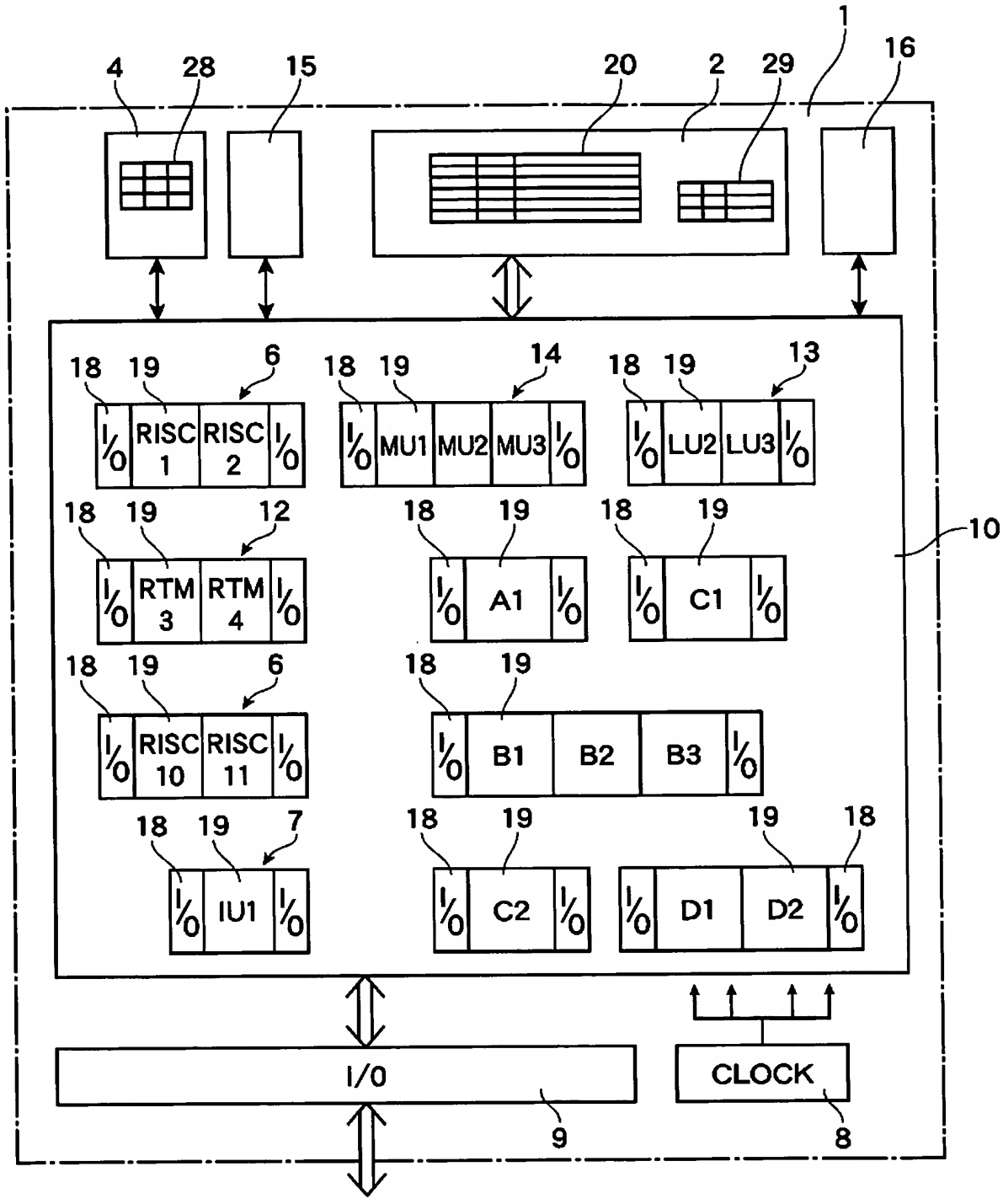
ラッチするレジスタを備えている、請求項31のデータ処理装置。

[34] 所定の数の前記エレメントにより構成される回路ブロックを複数備えている、請求項30のデータ処理装置。

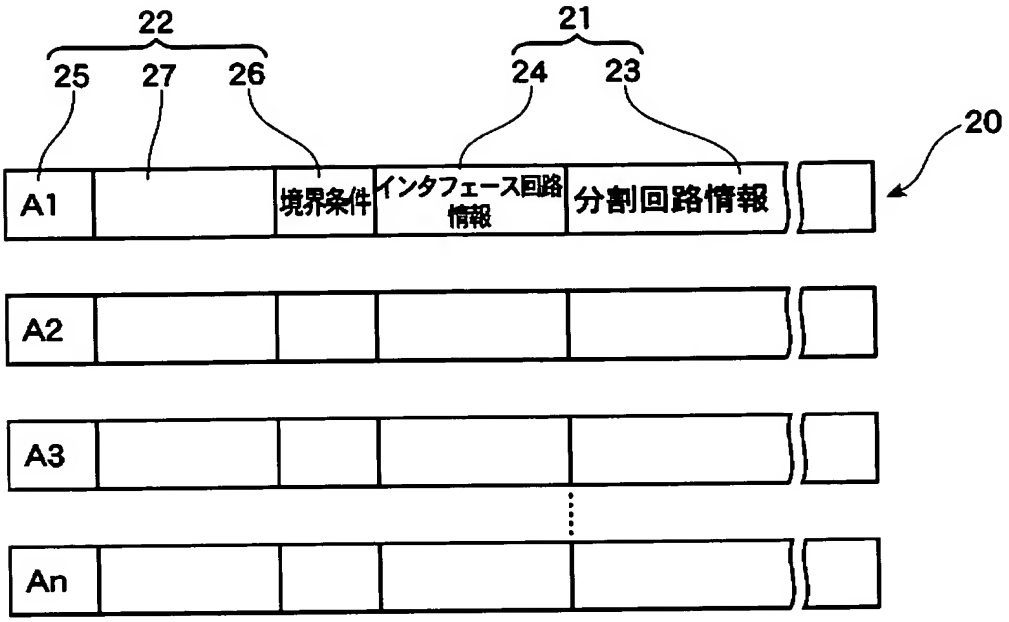
[図1]


1
20
2
RAM
4
5
15
11
RLC
13
LU
6
14
RISC
MU
18
19
19
10
12
I/O
A1
I/O
C1
I/O
18
RTM
I/O
B1
B2
B3
I/O
18
19
19
18
IU
I/O
C2
I/O
I/O
D1
D2
I/O
18
19
7
I/O
CLOCK
9
8

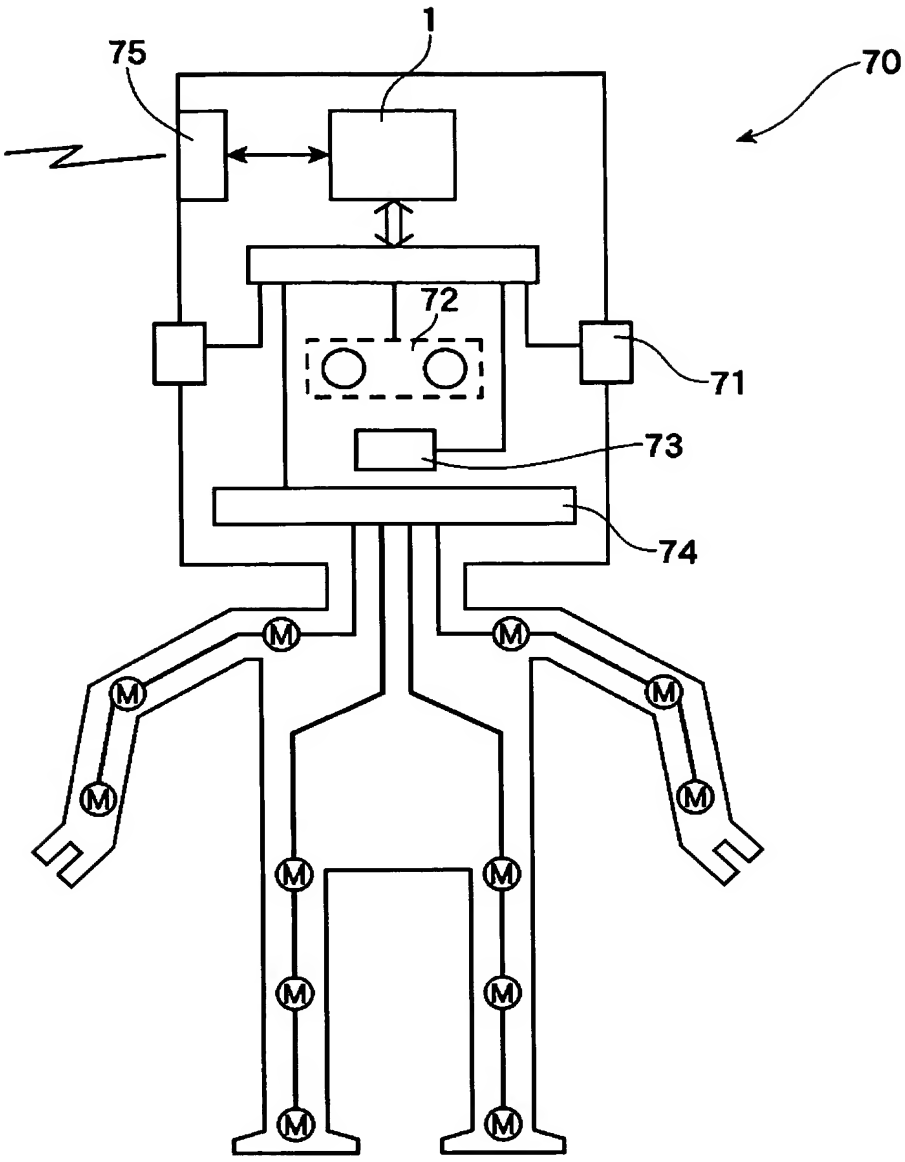
[図2]


1
4
28
15
20
2
29
16
18
19
6
14
13
I/O
RISC 1
RISC 2
MU1
MU2
MU3
LU2
LU3
12
RTM 3
RTM 4
A1
C1
10
RISC 10
RISC 11
B1
B2
B3
7
IU1
C2
D1
D2
I/O
CLOCK
9
8

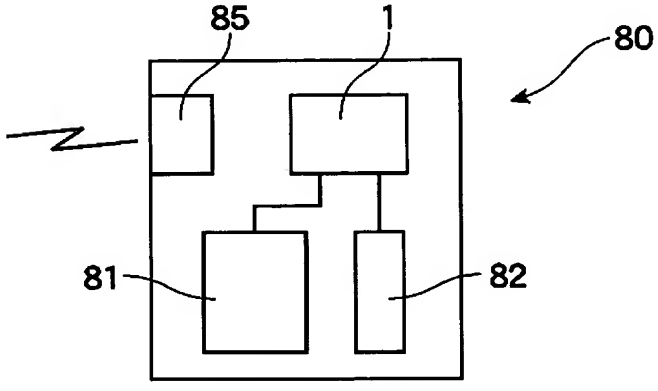
[図3]



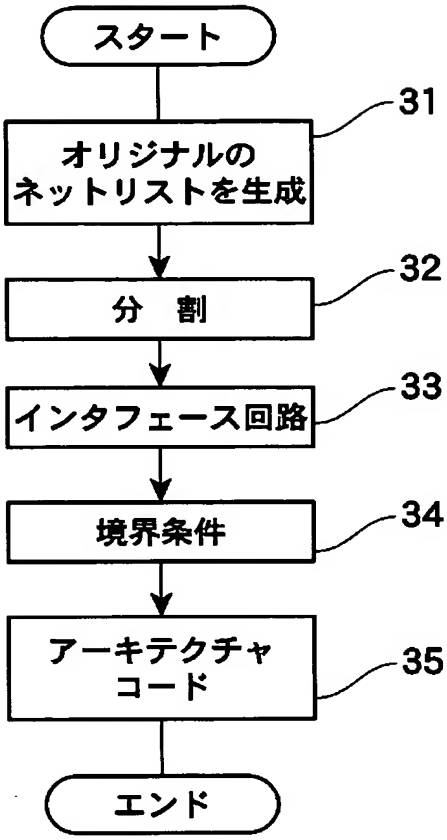
[図4]



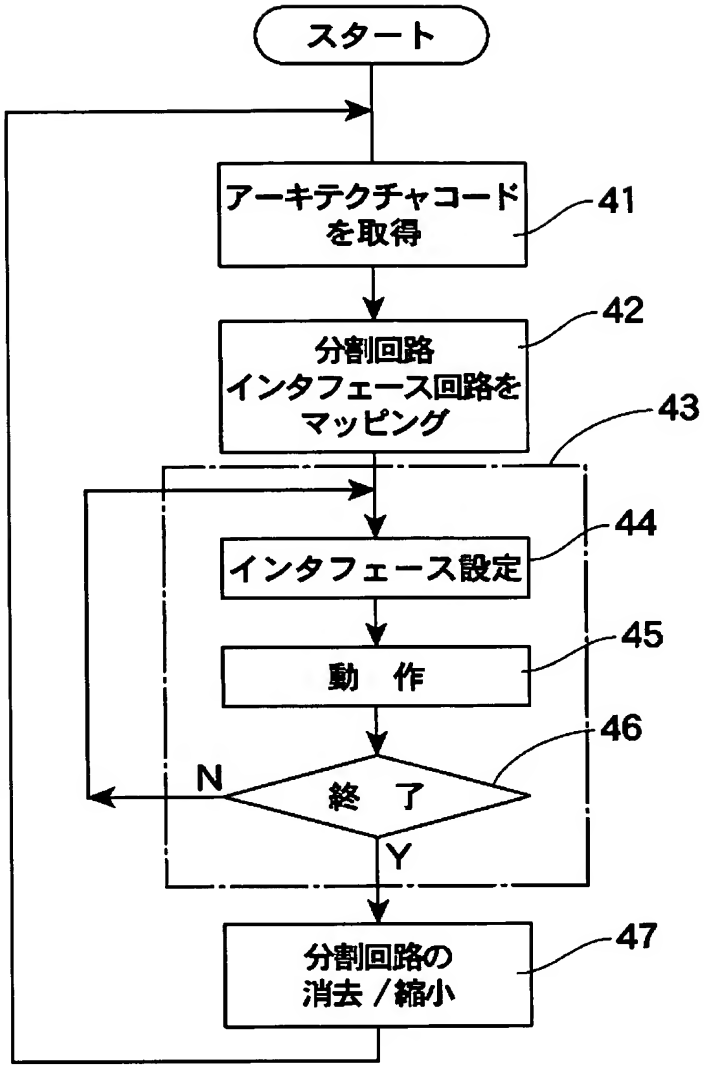
[図5]



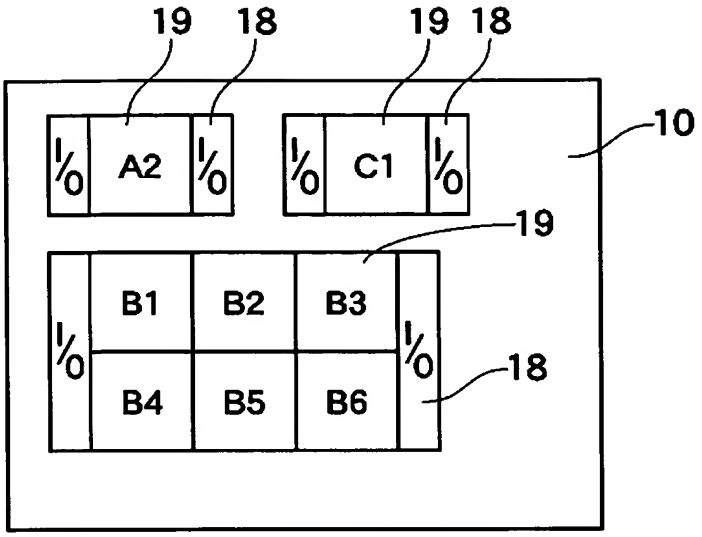
[図6]


スタート
オリジナルの
ネットリストを生成
31
分　割
32
インタフェース回路
33
境界条件
34
アーキテクチャ
コード
35
エンド

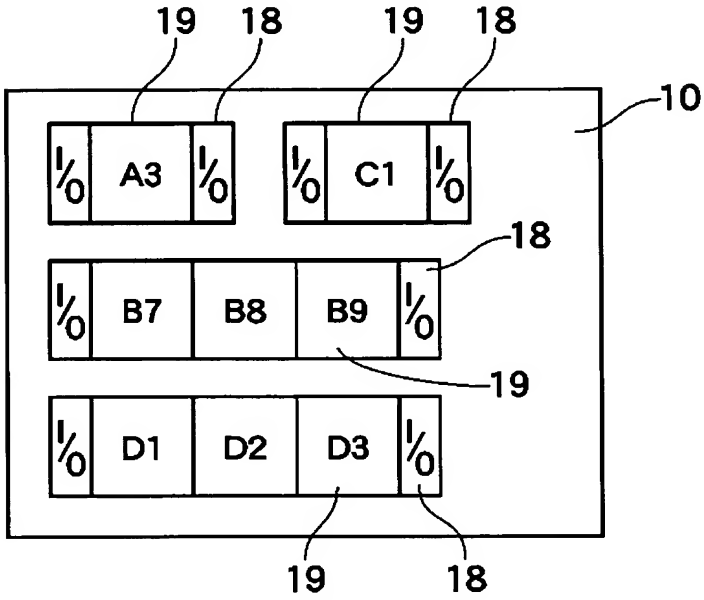
[図7]


スタート
アーキテクチャコードを取得
41
分割回路インタフェース回路をマッピング
42
43
インタフェース設定
44
動　作
45
終　了
46
N
Y
分割回路の消去／縮小
47

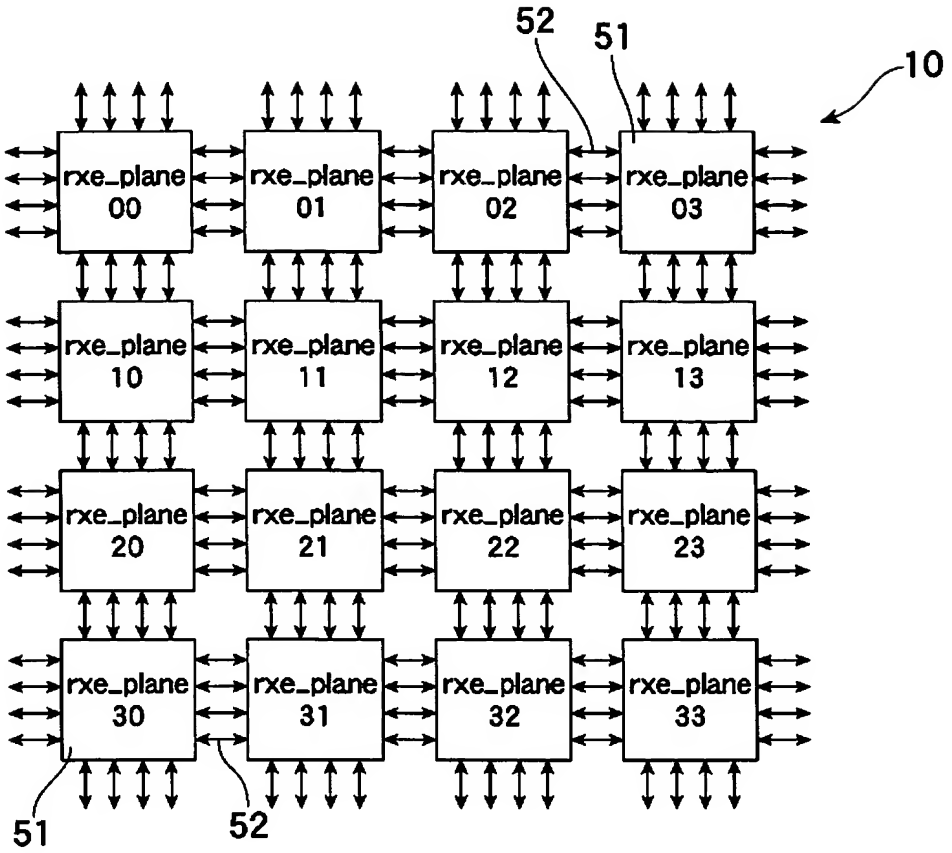
[図8]


19
18
19
18
I/O
A2
I/O
I/O
C1
I/O
10
19
B1
B2
B3
I/O
I/O
18
B4
B5
B6

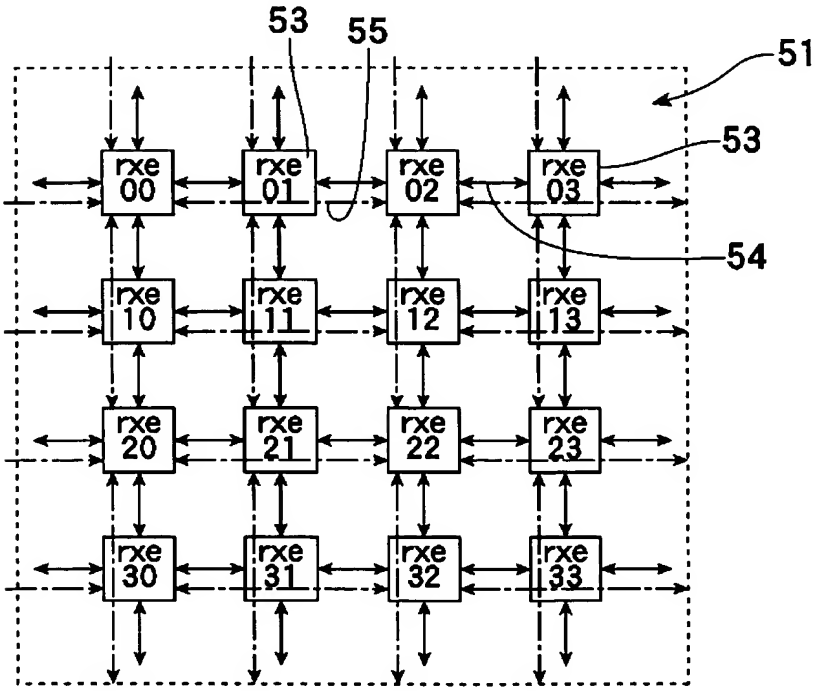
[図9]


19
18
19
18
I/O
A3
I/O
I/O
C1
I/O
10
18
I/O
B7
B8
B9
I/O
19
I/O
D1
D2
D3
I/O
19
18

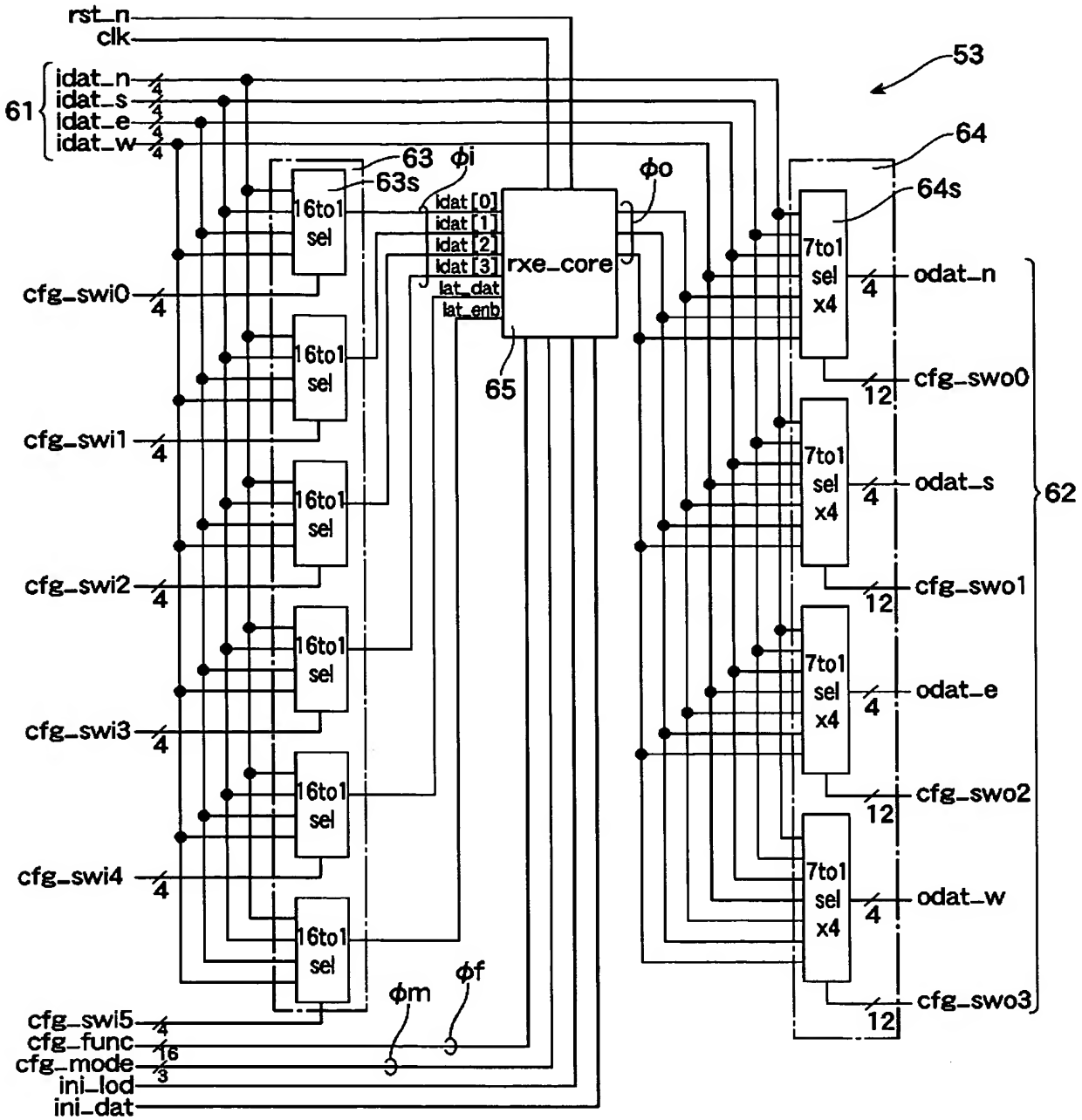
[図10]


52
51
10
rxe_plane 00
rxe_plane 01
rxe_plane 02
rxe_plane 03
rxe_plane 10
rxe_plane 11
rxe_plane 12
rxe_plane 13
rxe_plane 20
rxe_plane 21
rxe_plane 22
rxe_plane 23
rxe_plane 30
rxe_plane 31
rxe_plane 32
rxe_plane 33
51
52

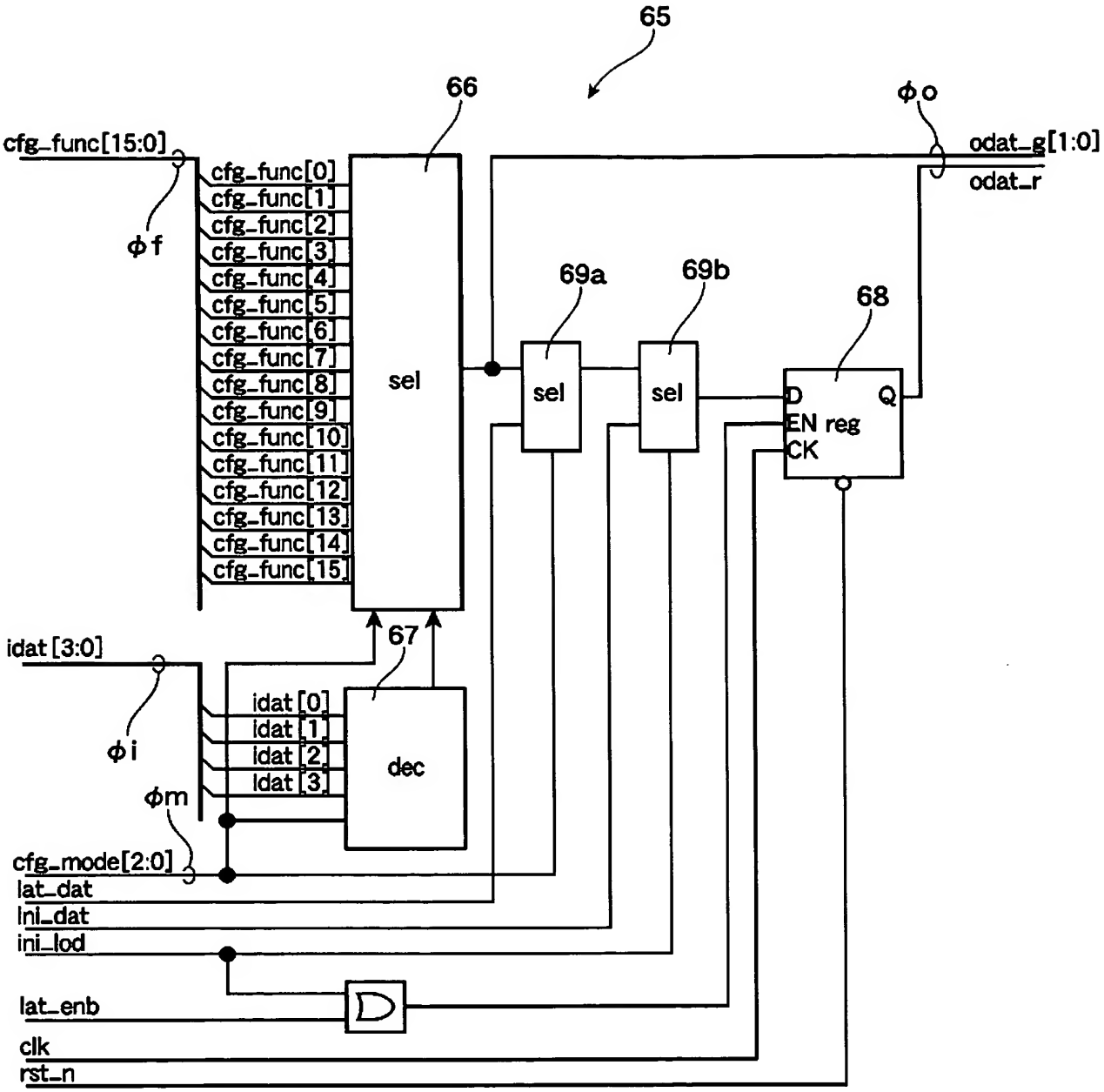
[図11]


53
55
51
53
54
rxe 00
rxe 01
rxe 02
rxe 03
rxe 10
rxe 11
rxe 12
rxe 13
rxe 20
rxe 21
rxe 22
rxe 23
rxe 30
rxe 31
rxe 32
rxe 33

[図12]


rst_n
clk
61
idat_n
idat_s
idat_e
idat_w
53
64
63
63s
φi
φo
64s
16to1 sel
idat [0]
idat [1]
idat [2]
idat [3]
lat_dat
lat_enb
rxe_core
65
7to1 sel x4
odat_n
odat_s
odat_e
odat_w
cfg_swo0
cfg_swo1
cfg_swo2
cfg_swo3
62
cfg_swi0
cfg_swi1
cfg_swi2
cfg_swi3
cfg_swi4
cfg_swi5
cfg_func
cfg_mode
ini_lod
ini_dat
φm
φf

[図13]


65
66
φo
cfg_func[15:0]
odat_g[1:0]
odat_r
cfg_func[0]
cfg_func[1]
cfg_func[2]
cfg_func[3]
cfg_func[4]
cfg_func[5]
cfg_func[6]
cfg_func[7]
cfg_func[8]
cfg_func[9]
cfg_func[10]
cfg_func[11]
cfg_func[12]
cfg_func[13]
cfg_func[14]
cfg_func[15]
φf
sel
69a
69b
68
sel
sel
D
EN reg
CK
Q
67
idat[3:0]
idat[0]
idat[1]
idat[2]
idat[3]
φi
dec
φm
cfg_mode[2:0]
lat_dat
ini_dat
ini_lod
lat_enb
clk
rst_n

[図14]

φm φi φo

cfg-mod [2:0]	idat [3]	[2]	[1]	[0]	odat_g [1]	[0]	odat_r	備考
000 (4in1out) [モード0]	0	0	0	0	0	cfg_func [0]	cfg-func [0]	odat-g [0] の値を保持
	0	0	0	1	0	cfg_func [1]	cfg-func [1]	
	0	0	1	0	0	cfg_func [2]	cfg-func [2]	
	0	0	1	1	0	cfg_func [3]	cfg-func [3]	
	0	1	0	0	0	cfg_func [4]	cfg-func [4]	
	0	1	0	1	0	cfg_func [5]	cfg-func [5]	
	0	1	1	0	0	cfg_func [6]	cfg-func [6]	
	0	1	1	1	0	cfg_func [7]	cfg_func [7]	
	1	0	0	0	0	cfg_func [8]	cfg_func [8]	
	1	0	0	1	0	cfg_func [9]	cfg_func [9]	
	1	0	1	0	0	cfg_func [10]	cfg_func [10]	
	1	0	1	1	0	cfg_func [11]	cfg_func [11]	
	1	1	0	0	0	cfg_func [12]	cfg_func [12]	
	1	1	0	1	0	cfg_func [13]	cfg_func [13]	
	1	1	1	0	0	cfg_func [14]	cfg_func [14]	
	1	1	1	1	0	cfg_func [15]	cfg_func [15]	
001 (4in1out) [モード1]	0	0	0	0	0	cfg_func [0]	lat_dat	レジスタ分離
	0	0	0	1	0	cfg_func [1]	lat_dat	
	0	0	1	0	0	cfg_func [2]	lat_dat	
	0	0	1	1	0	cfg_func [3]	lat_dat	
	0	1	0	0	0	cfg_func [4]	lat_dat	
	0	1	0	1	0	cfg_func [5]	lat_dat	
	0	1	1	0	0	cfg_func [6]	lat_dat	
	0	1	1	1	0	cfg_func [7]	lat_dat	
	1	0	0	0	0	cfg_func [8]	lat_dat	
	1	0	0	1	0	cfg_func [9]	lat_dat	
	1	0	1	0	0	cfg_func [10]	lat_dat	
	1	0	1	1	0	cfg_func [11]	lat_dat	
	1	1	0	0	0	cfg_func [12]	lat_dat	
	1	1	0	1	0	cfg_func [13]	lat_dat	
	1	1	1	0	0	cfg_func [14]	lat_dat	
	1	1	1	1	0	cfg_func [15]	lat_dat	

[図15]

cfg-mod [2:0]	idat [3]	idat [2]	idat [1]	idat [0]	odat_g [1]	odat_g [0]	odat-r	備考
010 (2in1out) [モード2]			0	0		cfg_func [0]	cfg_func [0]	上位2ビットと下位2ビットは独立した2系統になるレジスタは下位2ビットの結果を保持
			0	1		cfg_func [1]	cfg_func [1]	
			1	0		cfg_func [2]	cfg_func [2]	
			1	1		cfg_func [3]	cfg_func [3]	
	0	0			cfg_func [8]			
	0	1			cfg_func [9]			
	1	0			cfg_func [10]			
	1	1			cfg_func [11]			
011 (2in1out) [モード3]			0	0		cfg_func [0]		上位2ビットと下位2ビットは独立した2系統になるレジスタは上位2ビットの結果を保持
			0	1		cfg_func [1]		
			1	0		cfg_func [2]		
			1	1		cfg_func [3]		
	0	0			cfg_func [8]		cfg_func [8]	
	0	1			cfg_func [9]		cfg_func [9]	
	1	0			cfg_func [10]		cfg_func [10]	
	1	1			cfg_func [11]		cfg_func [11]	
100 (2in1out) [モード4]			0	0		cfg_func [0]	lat_dat	上位2ビットと下位2ビットは独立した2系統になるレジスタ分離
			0	1		cfg_func [1]	lat_dat	
			1	0		cfg_func [2]	lat_dat	
			1	1		cfg_func [3]	lat_dat	
	0	0			cfg_func [8]		lat_dat	
	0	1			cfg_func [9]		lat_dat	
	1	0			cfg_func [10]		lat_dat	
	1	1			cfg_func [11]		lat_dat	
101 (3in1out) [モード5]	×	0	0	0	cfg_func [8]	cfg_func [0]	cfg_func [0]	MSBは未使用 odat-g[0]の値を保持
		0	0	1	cfg_func [9]	cfg_func [1]	cfg_func [1]	
		0	1	0	cfg_func [10]	cfg_func [2]	cfg_func [2]	
		0	1	1	cfg_func [11]	cfg_func [3]	cfg_func [3]	
		1	0	0	cfg_func [12]	cfg_func [4]	cfg_func [4]	
		1	0	1	cfg_func [13]	cfg_func [5]	cfg_func [5]	
		1	1	0	cfg_func [14]	cfg_func [6]	cfg_func [6]	
		1	1	1	cfg_func [15]	cfg_func [7]	cfg_func [7]	
110 (3in1out) [モード6]	×	0	0	0	cfg_func [8]	cfg=func [0]	cfg_func [8]	MSBは未使用 odat-g[1]の値を保持
		0	0	1	cfg_func [9]	cfg_func [1]	cfg_func [9]	
		0	1	0	cfg_func [10]	cfg_func [2]	cfg_func [10]	
		0	1	1	cfg_func [11]	cfg_func [3]	cfg_func [11]	
		1	0	0	cfg_func [12]	cfg_func [4]	cfg_func [12]	
		1	0	1	cfg_func [13]	cfg_func [5]	cfg_func [13]	
		1	1	0	cfg_func [14]	cfg_func [6]	cfg_func [14]	
		1	1	1	cfg_func [15]	cfg_func [7]	cfg_func [15]	
111 (3in1out) [モード7]	×	0	0	0	cfg_func [8]	cfg_func [0]	lat_dat	MSBは未使用 レジスタ分離
		0	0	1	cfg_func [9]	cfg_func [1]	lat_dat	
		0	1	0	cfg_func [10]	cfg_func [2]	lat_dat	
		0	1	1	cfg_func [11]	cfg_func [3]	lat_dat	
		1	0	0	cfg_func [12]	cfg_func [4]	lat_dat	
		1	0	1	cfg_func [13]	cfg_func [5]	lat_dat	
		1	1	0	cfg_func [14]	cfg_func [6]	lat_dat	
		1	1	1	cfg_func [15]	cfg_func [7]	lat_dat	

[図16]

ファンクション	cfg-mode [2:0]	cfg-func [15:0]	備考
インバータ	010/011/100	xxxx_xxxx_xxxx_0101	最下位ビットを使用
2入力 AND	010/011/100	xxxx_xxxx_xxxx_1000	下位2ビットを使用
2入力 NAND	010/011/100	xxxx_xxxx_xxxx_0111	下位2ビットを使用
2入力 OR	010/011/100	xxxx_xxxx_xxxx_1110	下位2ビットを使用
2入力 NOR	010/011/100	xxxx_xxxx_xxxx_0001	下位2ビットを使用
2入力 EXOR	010/011/100	xxxx_xxxx_xxxx_0110	下位2ビットを使用
2入力 EXNOR	010/011/100	xxxx_xxxx_xxxx_1001	下位2ビットを使用
3入力 AND	101/110/111	xxxx_xxxx_1000_0000	下位3ビットを使用
3入力 NAND	101/110/111	xxxx_xxxx_0111_1111	下位3ビットを使用
3入力 OR	101/110/111	xxxx_xxxx_1111_1110	下位3ビットを使用
3入力 NOR	101/110/111	xxxx_xxxx_0000_0001	下位3ビットを使用
FullAdder	101/110/111	1110_1000_1001_0110	下位3ビットを使用 上位出力にキャリー 下位出力にサム
4入力 AND	000/001	1000_0000_0000_0000	
4入力 NAND	000/001	0111_1111_1111_1111	
4入力 OR	000/001	1111_1111_1111_1110	
4入力 NOR	000/001	0000_0000_0000_0001	
4入力 EXOR	000/001	0111_1111_1111_1110	
4入力 NOR	000/001	1000_0000_0000_0001	
AND_AND_OR	000/001	1000_1000_1000_1000	
AND_AND_NOR	000/001	0111_0111_0111_0111	
4入力 comparater(1111)	000/001	1000_0000_0000_0000	比較したい値を1にする

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2004/012380

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F9/30, H03K19/173

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F9/30, G06F15/78, G06F17/50, H03K19/173-19/177

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Toroku Jitsuyo Shinan Koho	1994-2004
Kokai Jitsuyo Shinan Koho	1971-2004	Jitsuyo Shinan Toroku Koho	1996-2004

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	JP 1-116690 A (Fujitsu Ltd.), 09 May, 1989 (09.05.89), (Family: none)	30,34 31-33
A	JP 2000-222447 A (NEC Corp.), 11 August, 2000 (11.08.00), (Family: none)	1-29
A	JP 7-503804 A (NATIONAL TECHNOLOGY, INC.), 20 April, 1995 (20.04.95), & WO 94/14123 A1	1-29
A	JP 2003-29969 A (TOKYO ELECTRON DEVICE LTD.), 31 January, 2003 (31.01.03), & WO 02/093404 A2	1-29

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search 10 November, 2004 (10.11.04)	Date of mailing of the international search report 22 November, 2004 (22.11.04)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (January 2004)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2004/012380

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2000-36738 A (Nippon Telegraph And Telephone Corp.), 02 February, 2000 (02.02.00), (Family: none)	30-34
A	JP 2000-138579 A (NEC Corp.), 16 May, 2000 (16.05.00), & US 6424171 B1	30-34

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2004/012380

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

The inventions of claims 1-29 relate to a method and a device for improving the use efficiency of a logic circuit area which can be reconfigured and exhibiting a maximum processing capability with a small amount of hardware resources.

The inventions of claims 30-34 relate to a device for rapidly exchanging logics without rewriting the LUT.

These inventions are not so linked as to form a single general inventive concept.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☒ No protest accompanied the payment of additional search fees.

国際調査報告	国際出願番号 PCT／JP2004／012380

A. 発明の属する分野の分類（国際特許分類（IPC））

Int. Cl⁷ G06F 9／30, H03K 19／173

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（IPC））

Int. Cl⁷ G06F 9／30, G06F 15／78, G06F 17／50
H03K 19／173 － 19／177

最小限資料以外の資料で調査を行った分野に含まれるもの
日本国実用新案公報 1922－1996年
日本国公開実用新案公報 1971－2004年
日本国登録実用新案公報 1994－2004年
日本国実用新案登録公報 1996－2004年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献のカテゴリー＊	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
X	JP 1－116690 A（富士通株式会社）1989. 05. 09（ファミリーなし）	30, 34
A		31－33
A	JP 2000－222447 A（日本電気株式会社）2000. 08. 11（ファミリーなし）	1－29
A	JP 7－503804 A（ナショナル テクノロジー インコーポレイテッド）1995. 04. 20 & WO 94／14123 A1	1－29

☒ C欄の続きにも文献が列挙されている。 ☐ パテントファミリーに関する別紙を参照。

＊ 引用文献のカテゴリー
「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
10. 11. 2004	22.11.2004

国際調査機関の名称及びあて先	特許庁審査官（権限のある職員）		
日本国特許庁（ISA／JP） 郵便番号100－8915 東京都千代田区霞が関三丁目4番3号	後藤 彰	5B	4226
	電話番号 03－3581－1101 内線 3545		

様式PCT／ISA／210（第2ページ）（2004年1月）

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２００４／０１２３８０

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ａ	ＪＰ　２００３－２９９６９　Ａ（東京エレクトロンデバイス株式会社）２００３．０１．３１ ＆　ＷＯ　０２／０９３４０４　Ａ２	１－２９
Ａ	ＪＰ　２０００－３６７３８　Ａ（日本電信電話株式会社）２０００．０２．０２（ファミリーなし）	３０－３４
Ａ	ＪＰ　２０００－１３８５７９　Ａ（日本電気株式会社）２０００．０５．１６　＆　ＵＳ　６４２４１７１　Ｂ１	３０－３４

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２００４／０１２３８０

第Ⅱ欄　請求の範囲の一部の調査ができないときの意見（第１ページの２の続き）

法第８条第３項（ＰＣＴ１７条(2)(a)）の規定により、この国際調査報告は次の理由により請求の範囲の一部について作成しなかった。

1．☐ 請求の範囲 _____ は、この国際調査機関が調査をすることを要しない対象に係るものである。
つまり、

2．☐ 請求の範囲 _____ は、有意義な国際調査をすることができる程度まで所定の要件を満たしていない国際出願の部分に係るものである。つまり、

3．☐ 請求の範囲 _____ は、従属請求の範囲であってＰＣＴ規則6.4(a)の第２文及び第３文の規定に従って記載されていない。

第Ⅲ欄　発明の単一性が欠如しているときの意見（第１ページの３の続き）

次に述べるようにこの国際出願に二以上の発明があるとこの国際調査機関は認めた。

請求の範囲１－２９は、再構成可能な論理回路領域の利用効率を向上でき、少ないハードウェアリソースで最大限の処理能力を発揮する方法及び装置に関するものである。
請求の範囲３０－３４は、ＬＵＴを書き換えることなく高速に論理を交換する装置に関するものである。
これらは、単一の一般的発明概念を形成するように連関しているものではない。

1．☒ 出願人が必要な追加調査手数料をすべて期間内に納付したので、この国際調査報告は、すべての調査可能な請求の範囲について作成した。

2．☐ 追加調査手数料を要求するまでもなく、すべての調査可能な請求の範囲について調査することができたので、追加調査手数料の納付を求めなかった。

3．☐ 出願人が必要な追加調査手数料を一部のみしか期間内に納付しなかったので、この国際調査報告は、手数料の納付のあった次の請求の範囲のみについて作成した。

4．☐ 出願人が必要な追加調査手数料を期間内に納付しなかったので、この国際調査報告は、請求の範囲の最初に記載されている発明に係る次の請求の範囲について作成した。

追加調査手数料の異議の申立てに関する注意
☐ 追加調査手数料の納付と共に出願人から異議申立てがあった。
☒ 追加調査手数料の納付と共に出願人から異議申立てがなかった。

第Ⅳ欄　要約（第１ページの５の続き）

本発明においては、アプリケーションを実行するための回路の少なくとも一部であるオブジェクト回路を動的に再構成可能な論理回路の一部にマッピングするためのオブジェクト回路情報（２３）と、オブジェクト回路に接するインタフェース回路を論理回路にマッピングするためのインタフェース回路情報（２４）と、インタフェース回路において実現する境界条件（２６）とを含むアーキテクチャコード（２０）を使用する。

本発明のデータ処理装置は、アーキテクチャコード（２０）を取得するロードユニットと、アーキテクチャコードのオブジェクト回路情報（２３）及びインタフェース回路情報（２４）により、論理回路領域にオブジェクト回路とインタフェース回路をマッピングするマッピングユニットと、アーキテクチャコードの境界条件（２６）にしたがってインタフェース回路を制御する動作制御ユニットを有する。